Draft version June 7, 2024
Typeset using LaTeX **twocolumn** style in AASTeX631

CMB Research - Spring 2024 Progress Report

Emma Bacarra[1]

[1] *Advisor: Nima Sedaghat*
University of Washington Department of Astronomy

1. INTRODUCTION

Within a microsecond of the Big Bang, the powerful release of hot energy created particles of matter (electrons, neutrons, and protons) and light (photons). Most electrons at this point weren't bound to an atomic nucleus yet, which meant that they were knocking into other particles like photons. Called the cosmic fog, looking at the Universe during this period appears blurry from the electrons interrupting the path of photons trying to travel towards us.

It wasn't until 380,000 years later that the Universe cooled enough for atomic nuclei and electrons to combine into Hydrogen and Helium - a period known as recombination. The Cosmic Microwave Background (CMB) is a clear, 3-D snapshot of the Universe from that first moment when microwave photons could travel interruption-free: also known as decoupling (NASA (2024)).

The CMB is an important source of information that can tell us about the features of the early universe - for example, studying it could help prove the theory of inflation. Inflation addresses some anomalies in our knowledge of the Universe that otherwise couldn't be explained - right before the Big Bang, the Universe expanded faster than the speed of light and produced a background of gravitational waves. It's unknown how the Universe gained momentum for such exponential growth; moreover, those gravitational waves are too faint to directly measure which is where the CMB polarization comes in (har (2024)).

When the photons became free to travel after recombination, the direction they traveled was determined by their last collision with an unbound electron (ESA/-Planck (2015)). Known as polarization, this preferred direction reveals distinctive patterns that are associated with particular types of nearby influence. E-mode polarization describes the perpendicular (and parallel) influence of tiny density fluctuations in matter, mapping the stage for where today's stars & galaxies formed. B-mode polarization refers to a pattern of 45° angles (from the direction of propagation) created by the gravitational waves produced from inflation (Bischoff (2016)). By searching for and studying B-mode signals, that could help prove the theory of inflation; however, for such highly sensitive measurements, it's easy for data to get contaminated by noise like dust or synchrotron radiation mentioned in Stivoli's paper.

To escape dust and synchrotron noise, Stivoli et al. (2010) explored two methods to isolate B-modes: balloon-based observations and ground-based observations. For the ground-based method, the atmosphere made it difficult to clean up data and isolate B-mode signals; for that same reason, the balloon was able to escape atmospheric interference which allowed for operating at higher frequencies (thus eliminating synchrotron radiation). However, both methods were able to deduce variations in CMB polarization through parameter estimation with statistical techniques applied to WMAP (2012) data.

In a separate group of research, by Eriksen et al. (2006), they addressed the issue of isolating B-mode signals by suggesting a more direct method to what's known as the component separation problem. To record a "pure" signal of CMB radiation, the data must be void of noise which can come in the form of dust, objects, galactic emissions, or anything that doesn't belong to the CMB foreground. For important signals, they can be fitted to parametric models while the remaining free parameters can be fitted using Monte Carlo and nonlinear algorithms. What's beneficial about this approach is that the data is scalable and the method skips the step of feature extraction.

A third group of research, by Bobin, J. et al. (2013), suggested using the method of Generalized Morphological Component Analysis (GMCA). It takes advantage of sparsity to generalize a model, making it easier to sort out unnecessary signals. It also experimented with Independent Component Analysis (ICA) and Internal Linear Combination (ILC) methods. Using Planck/ESA (2006) data, each of these methods were evaluated for how accurately they could retrieve the CMB signal from noisy, nonlinear data. Of all the techniques, GMCA performed the best for beam variations across frequencies and variation in emission line spacing.

1.1. *Definitions in CMB Research*

To catch up to speed on relevant concepts and vocabulary for the CMB, these definitions were noted:

- **Decoupling**: the first moment microwave photons could travel interruption-free of unbound electrons
- **Surface of Last Scattering**: a "shell" or "surface" at the right distance in space where the photons from decoupling can be received

$$C(\theta) = \frac{l}{4\pi} \sum_{l=0}^{\infty} (2l+1) C_l P_l \cos(\theta) \tag{1}$$

- **Correlation Function** $C(\theta)$: statistical measure of total temperature fluctuation (Pettini (2018))
  - $C(\theta)$ compares the temperature fluctuations between two points on the SoLSc separated by an angle θ
  - The result is an average of all the products between temperature fluctuations of a pair separated by θ
  - l: Multipole moment
  - C_l: The multipole components of the Legendre Polynomial expansion P_l
- **Angular Power Spectra**

$$C_l^{XY,data} = \frac{l}{2l+1} \sum_{m=-l}^{l} a_{X,lm} a_{Y,lm}^{*} \tag{2}$$

  - X, Y can be represented by T, E, or B
  - a represents the spin-2 coefficient of a polarization mode
- **TT (Temperature)**: spectrum data derived from the following values at each l
  - Power Spectrum, given in units of μK^2

$$\frac{l}{2\pi} (l+1) C_l^{TT,data} \tag{3}$$

  - Noise: $\frac{l}{2\pi} (l+1) N_l$
  - Effective Sky Fraction $f_{sky,l}$
  - Best Fit Theory Spectrum $C_l^{TT,th}$
- **TE**: temperature-E-mode cross-power spectrum (temperature-polarization spectrum)
- **EE**: E-mode polarization angular auto-power spectrum

TB, EB, and BB are expected to be zero - in general, they can be non-zero from data contamination (foreground signals)

2. SPECTRAL DATA

We began with an exploration of potential datasets to analyze, starting with the data products from WMAP, ACT, and Planck on the LAMBDA database. However, some of the data products explored from ACT and Planck may not be helpful for this project - their spectral data was not on the same scale, making it difficult to compare. ACT's database provided the multipole moment and temperature, however the error was provided for binned values - Planck also had a similar feature. While those could be useful for other areas of research, unbinned values may be more useful in this project.

2.1. *WMAP Overview*

WMAP became the focus, which includes the temperature and polarization measurements of the CMB. 5 broadband categories are provided in the database, where each has at least two wavelengths of its kind. The W-band has four provided wavelengths.

Table 1: Mission Characteristics

	K	Ka	Q	V	W
Wavelength (mm)	13	9.1	7.3	4.9	3.2
Frequency (GHz)	23	33	41	61	94
Bandwidth (GHz)	5.5	7.0	8.3	14.0	20.5
Differencing Assemblies	1	1	2	2	4
Radiometers	2	2	4	4	8
Channels	4	4	8	8	16
Beam Size (deg)	0.88	0.66	0.51	0.35	0.22
Sys. Temperature (K)	29	39	59	92	145
Sensitivity (mK $\sec^{1/2}$)	0.8	0.8	1.0	1.2	1.6

The WMAP database provides nine years of research in total, released to the public in five phases. We started by looking at data from the fifth and final release.

Table 2: Release Characteristics

Data Release	1st	2nd	3rd	4th	5th
Date	02/03	03/06	03/08	01/10	06/12
Years of Data	1	3	5	7	9
Temp. Analysis	Y	I	I	I	I
Polzn. Analysis	N	Y	I	I	I
Sequential Data	N	Y	N	N	N
Error Analysis	N	I	I	I	I
Map Algorithm	N	C	N	N	N
Calib Algorithm	N	N	I	I	I
Beam Maps	N	N	I	I	I

Y: yes, N: no, I: improved, C: changed

2.2. *Understanding WMAP's Data*

First, a few definitions were noted:

- **COBE Sky Map**: uncorrected dipole temperature shift across the sky
- **Radiometric Gain**: proportionality constant between temperature difference (input) & output voltage (output)

WMAP's microwave system consists of ten 4-channel differencing assemblies (or broadbands). For each assembly, the signals are obtained from a pair of radiometers tuned to their frequency. The output signal is proportional to temperature (brightness) difference between the two radiometers.

Due to random asymmetries, the assemblies can have noisy error. The temperature of the CMB also varies, being 6.706 mK brighter in one sky direction than the opposite because of the Doppler Effect. To calibrate the signals, known microwave sources are used to subtract the noise. Raw output data is compared to the expected signal from known dipole anisotropies then calibrated to true temperature differences in the sky.

The 5th version of WMAP's 9-year data temperature power spectrum was plotted to get a better understanding of the data - it is unbinned and contains the following columns:

1. The multipole moment l
2. The power spectrum in units of μK^2
3. The error derived from the diagonal elements of the covariance matrix in units of μK^2
4. The portion of column 3 attributed to measurement errors in units of μK^2
5. The portion of column 3 attributed to cosmic variance in units of μK^2, assuming the best-fit λCDM model.

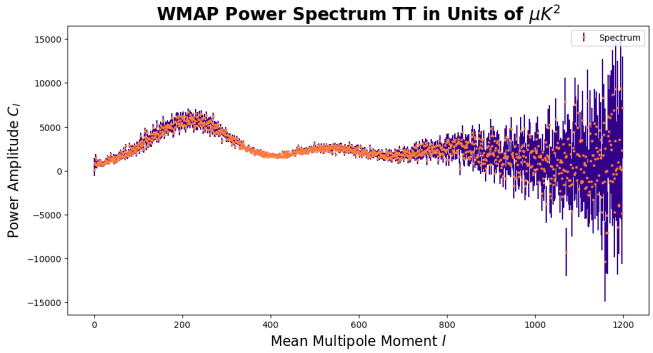

Figure 1: The plotted TT (temperature) power spectrum from WMAP's 9-year data.

3. TEMPERATURE SKY MAPS

Following the initial plot of the temperature power spectrum, we proceeded to mapping the existing temperature sky maps in order to gain familiarity with WMAP's data. Like the temperature power spectrum plotted previously, the chosen sky map product is also from data release 5: the Single Year Temperature Res 9 Sky Maps Per Differencing Assembly, or the full resolution Intensity (I) sky map for each broadband frequency from year 1 to 9. All provided broadbands from WMAP were explored: K1, Ka1, Q1, Q2, V1, V2, W1, W2, W3, and W4. Figure 2 is an example of how the images were plotted - a comparison was done between all years for all broadband wavelengths.

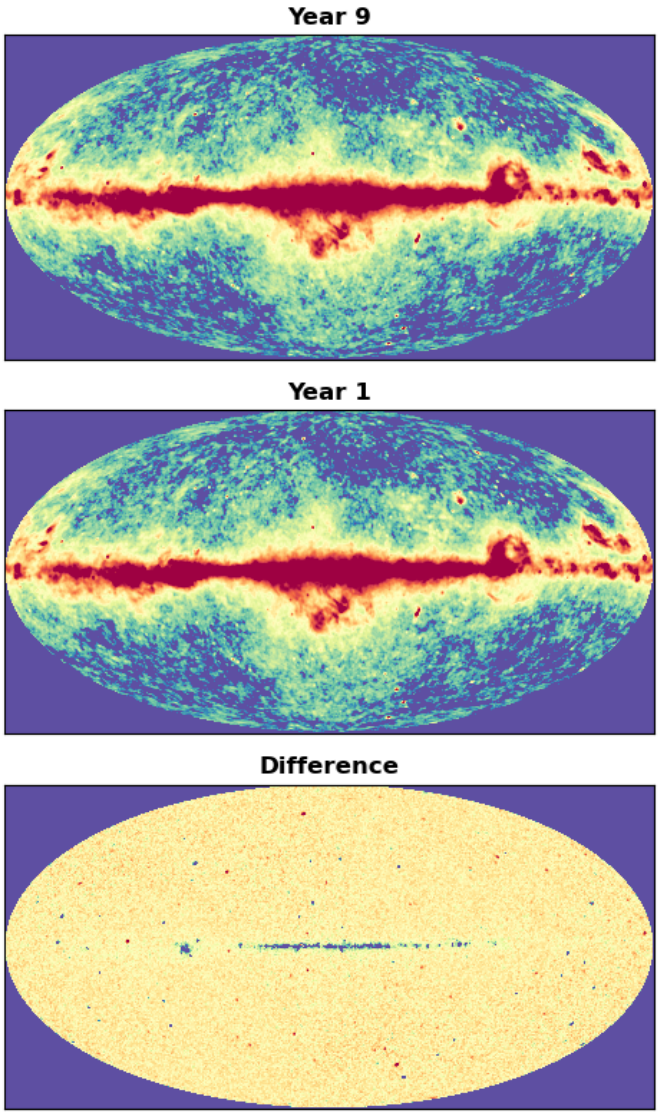

Figure 2: Difference between Year 9 and Year 1

4. NEURAL NETWORK ARCHITECTURES

In classical machine learning, models can be used as a way of streamlining huge processes - however, these models can only handle smaller sets of well-structured data. It can be a task as simple as linear regression, but the important idea is that an algorithm is learning these relationships on its own. Classical machine learning algorithms require explicit programming to learn from a dataset - that is, the features or properties of a dataset must be manually identified before training algorithms. If there's any form of non-linearity in the dataset, it can be quite difficult to extract features. When doing this step of feature extraction manually, it also poses a risk of overlooking or losing valuable information about a property or its relationship to others. Handling non-linearity is made easy with deep learning neural networks.

4.1. *Introduction*

A subset of machine learning, deep learning is much more powerful in terms of what it can handle. What's different is the presence of neurons, much like the actual functionality of a human brain (see figure 3).

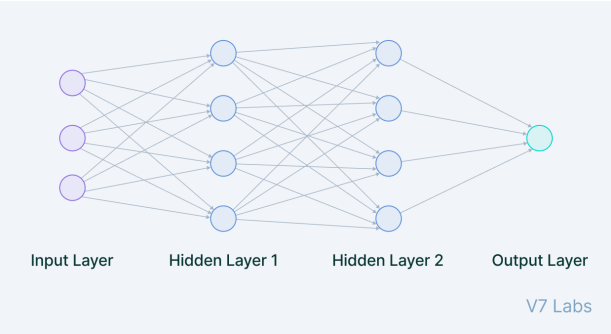

Figure 3: Simplified diagram of neural network architecture components, via Baheti (2021)

Each "circle" is called a neuron, which represents a matrix that manipulates the input upon receiving it. Just like the brain, these neurons interact with each other: their connections allow them to pass along information to each other in what's called a connectivity matrix. The matrix relays the weight, or importance, of information and is one of the many parameters that get adjusted by the model during a training session. Through a system of trial and error, the neural network learns to set the proper weights in order to achieve the targeted outcome. There can be numerous amounts of neurons in these models, making it capable of handling millions of data points at a time - classical machine learning models can only handle up to thousands of data points.

There are several methods to select to guide the neural network during a training process. Optimization is the idea of learning through trial and error - when a neural network takes in a data set, it makes note of the parameter settings that helped (or hindered) it from reaching a target output. The next time it takes in the data set again, the neural network considers what it learned from the previous cycles to define new parameter settings. A successful training session means that the neural network is able to output answers very close to the target.

Neural network models can serve several different purposes, from classification, to language translation, and image analysis. They are well integrated into society - from autopilot cars, to weather forecasting, and social media algorithms. Their strong capabilities make for amazing tools, however common ones in every day life are known as supervised learning models. They require a ground truth after a training session as a way of gathering a baseline for performance. But what if we could program a model to train *without* the limits of supervision?

4.2. *What is an autoencoder?*

As an unsupervised learning model, an autoencoder is a type of neural network that is designed for predictive or generative tasks. Like any deep learning model, it's designed to identify patterns for a set of information. The key difference is that it is capable of identifying common factors without the pressure of producing a desired output. Without the limitations of specific directions, an autoencoder can independently learn any pattern that occurs naturally in the dataset.

Autoencoders fall within the category of Encoder/Decoder neural network architectures, who create numerical representations of the inputs that map to specific meanings. Similar to an identification number or license plate, the input becomes represented by a multi-dimensional latent vector, where each element, or latent variable, relays a type of significance or relationship between the elements of other input vectors (de Kleut (2020)). These latent variables are not directly observable, but they impact the interpretation of how data is fundamentally distributed. They exist in a latent space, which represents the most important pieces of information from the data, and during a training session the model will learn which latent variables are most important (Bergmann (2023)). Using this information, the autoencoder attempts to reconstruct the original data and uses that comparison as an evaluation of its performance.

4.3. *Variational Autoencoders*

Autoencoders compress data into a fixed (linear) latent space, which can cause data points of a group to vary (nonlinear) after compression. When mapping these numerical representations, that means that there can be outliers from a group, which prevents an autoencoder from generating an output that can also map back to the same group. This is useful for cases where you want to preserve more information by keeping everything unnormalized, like making exact (or perhaps almost exact) predictions (Anwar (2021)).

Variational autoencoders are similar, except they normalize the latent space to remove outliers. Thanks to the Kulback-Leibler divergence, or the measure of how one probability distribution $P(A)$ differs from another $P(B)$, this is useful for cases an autoencoder can't do such as generating meaningful outputs (Rocca (2019)). The KLD is used as a normalization term to calculate the loss.

The difference between them is the output of the encoder - in an autoencoder, the encoder layer outputs the compressed latent data, while a the encoder layer of a VAE outputs the mean μ and the variance (the average squared deviations from the mean) σ of each latent data point. The variance is represented by a covariance matrix, which is essentially the variance of each element represented in a diagonal. After the encoder, there's an additional step to get a new representation of the original input data - the reconstruction term, which uses probability based on the μ and σ to improve accuracy, is used in combination with the KLD normalization term to generate a sample of the data's latent representation. Then, it is passed through a decoder.

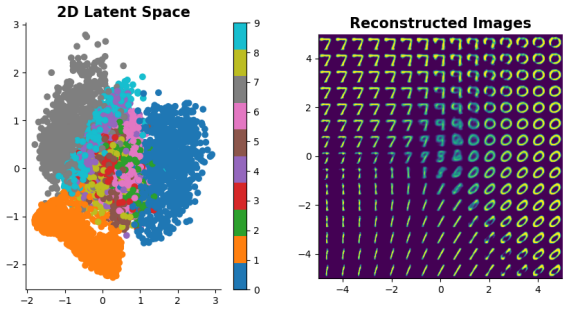

Figure 4: Left: 2D Latent space representation. Right: Reconstruction of images along points in a segment of the latent space. The VAE performance shown has room for improvement, but the general idea is that VAEs can normalize nonlinear data into the latent space, allowing for reproducibility. The grey, orange, and dark blue regions are good examples of normalizing to a linear space, as they all group together into the same region.

4.4. *Preliminary Exercises*

Inspired by Sedaghat et al. (2021), we follow the variational autoencoder architecture used in *Machines Learn to Infer Stellar Parameters* as an example for how this project will carry out an exploration of the temperature maps of the CMB. To gain a sense of familiarity with this architecture, a few novel examples - from Kalantar (2022) and Lee (2018) - were reviewed using the MNIST dataset along with supplemental reading.

The difference between the architectures of the two examples is that the first example uses leaky ReLU activation functions, meaning that it can handle nonlinearity in the dataset. The second example only uses fully connected layers (linear), which means the dimensions are preserved throughout the forward passing. Another difference is their reparameterization process on the encoder outputs before passing to the decoder. Their processes are similar, though the logarithmic variance varies due to their abilities of handling nonlinearity. The first example, which handles nonlinearity, passes the variance directly to torch.randn_like() before reparametizing. The second example, which doesn't handle nonlinearity, calculates the standard deviation by raising the encoder outputs to half the variance. It then passes that as the argument into torch.randn_like() to reparameterize the encoder outputs. Besides that, their implementations of the random-like matrix on the encoder outputs are the same, taking the sum of the product of the random matrix & epsilon with the encoder output mean.

Similarly, both examples use the same optimizer and loss functions. The reparameterization layer, which generates the latent sample of a data point, generates a random like matrix of the same dimensions as the input parameter. The elements of the matrix are selected to calculate to a mean of 0 and variance of 1. It is then multiplied by the standard deviation, which is a matrix of outputs raised to an exponent of half the variance, and added together with the mean produced by the encoder layer. Both examples also chose Adam as their optimizer, which uses stochastic optimization and applies an adaptive learning rate based on previous iterations. A custom loss function in both is centered around the Kulback-Leibler divergence loss, to measure the difference between two probability distributions. In partnership with binary cross entropy, the difference between predicted binary outcomes and actual binary labels is comparec and innaccurate predictions are penalized.

Once familiar with autoencoders at a comfortable level, an initial version of variational autoencoder architecture was developed to practice its implementation towards the research project (see figure **??**).

```
# building the encoder class
class encoder(nn.Module):
    def __init__(self, input_dim, hidden_dim, num_layers, latent_dim, leak=0.2, drop=0.5):
        super(encoder, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.LeakyReLU(leak),
            nn.Dropout(drop),
            nn.Linear(hidden_dim, num_layers),
            nn.LeakyReLU(leak),
            nn.Dropout(drop)
            )

        # latent mean and covariance matrix (diagonal matrix of variances)
        self.latent_mean = nn.Linear(num_layers, latent_dim)
        self.latent_variance = nn.Linear(num_layers, latent_dim)

    def forward(self, x):
        x = self.encoder(x)
        mean, variance = self.latent_mean(x), self.latent_variance(x)
        return mean, variance

# building the decoder class
class decoder(nn.Module):
    def __init__(self, input_dim, hidden_dim, num_layers, latent_dim, leak, drop):
        super(decoder, self).__init__()

        self.decoder = nn.Sequential(
                nn.Linear(latent_dim, num_layers),
                nn.LeakyReLU(leak),
                nn.Dropout(drop),
                nn.Linear(num_layers, hidden_dim),
                nn.LeakyReLU(leak),
                nn.Dropout(drop),
                nn.Linear(hidden_dim, input_dim),
                nn.Sigmoid()
                )# output dimensions are the same as original inputs

    def forward(self, x):
        return self.decoder(x)

# assembling the VAE class
class vae(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, num_layers, latent_dim, leak, drop):
        super(vae, self).__init__()

        self.encoder = encoder(
                                input_dim=input_dim,
                                hidden_dim=hidden_dim,
                                num_layers=num_layers,
                                latent_dim=latent_dim,
                                leak=leak, drop=drop
                            ).to(device)
        self.decoder = decoder(
                                input_dim=input_dim,
                                hidden_dim=hidden_dim,
                                num_layers=num_layers,
                                latent_dim=latent_dim,
                                leak=leak, drop=drop
                            ).to(device)

    # sample z based on its latent space distribution
    def reparameterize(self, mean, variance):
        stdev = torch.randn_like(variance).to(device)
        return mean + variance*stdev # <-- z reparameterized

    def forward(self, x):
        x = x.to(device)
        mean, variance = self.encoder(x)
        z_sample = self.reparameterize(mean, variance)
        z = self.decoder(z_sample) # reconstruct the original input
        return z, mean, variance
```

5. LOOKING AHEAD

To work towards characterizing the CMB, we will continue to use the variational autoencoder architecture as a means for exploring temperature maps. With an unsupervised learning model, there is potential for it to make an inference of familiar physics on its own. Perhaps, the model will even be able to recognize an unknown property or relation of the data. We look forward to uncovering what the variational autoencoder may discover.

For further discussion, the following questions will be kept in mind: 1) Is there potential significance in categorizing the sky map data by broadband? 2) Where can the VAE be improved to optimize an unsupervised analysis of the CMB? 3) Of the maps available from WMAP (Stokes I (temperature), Stokes Q and U (Polarization), and bandpass mismatch (S map)), which is the best source for training data?

REFERENCES

2024, Center for Astrophysics - Harvard & Smithsonian. https://www.cfa.harvard.edu/research/topic/cosmic-microwave-background

Anwar, A. 2021, Medium: Towards Data Science. https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2

Baheti, P. 2021, V7. https://www.v7labs.com/blog/neural-network-architectures-guide

Bergmann, D. 2023, IBM. https://www.ibm.com/topics/autoencoder

Bischoff, C. 2016, Center for Astrophysics - Harvard & Smithsonian. https://lweb.cfa.harvard.edu/~cbischoff/cmb/

Bobin, J., Starck, J.-L., Sureau, F., & Basak, S. 2013, AA, 550, A73, doi: 10.1051/0004-6361/201219781

de Kleut, A. V. 2020. https://avandekleut.github.io/vae/

Eriksen, H. K., Dickinson, C., Lawrence, C. R., et al. 2006, The Astrophysical Journal, 641, 665–682, doi: 10.1086/500499

ESA/Planck. 2015, NASA Jet Propulsion Laboratory. https://www.jpl.nasa.gov/images/pia18916-polarization-of-the-cosmic-microwave-background

Kalantar, R. 2022, Medium. https://medium.com/@rekalantar/variational-auto-encoder-vae-pytorch-tutorial-dce2d2fe0f5f

Lee, A. 2018, GitHub. https://github.com/lyeoni/pytorch-mnist-VAE/tree/master

NASA. 2024, NASA For Science. https://science.nasa.gov/universe/overview/

Pettini, M. 2018, Introduction to Cosmology. https://people.ast.cam.ac.uk/~pettini/Intro%20Cosmology/Lecture10.pdf#page=3

Planck/ESA. 2006

Rocca, J. 2019, Medium: Towards Data Science. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Sedaghat, N., Romaniello, M., Carrick, J. E., & Pineau, F.-X. 2021, Monthly Notices of the Royal Astronomical Society, 501, 6026–6041, doi: 10.1093/mnras/staa3540

Stivoli, F., Grain, J., Leach, S. M., et al. 2010, Monthly Notices of the Royal Astronomical Society, 408, 2319, doi: 10.1111/j.1365-2966.2010.17281.x

WMAP. 2012, WMAP Data Release 5. https://lambda.gsfc.nasa.gov/product/wmap/current/