Technical University of Denmark

DTU Compute

Developing a Convolutional Variational Auto-Encoder for the ASIM Dataset

Rasmus Christian Jørgensen (s164044)

Supervisor(s): Ole Winther

May 10, 2021

Rasmus C. Jørgensen 10/05-2021

Abstract

In this report a CVAE for grouping observations of TLEs from the MMIA instrument onboard ASIM has been developed. The goal being to ease the manual labor when reviewing data from the instrument. The optimization of the CVAE builds on the CVAE presented in [Jørgensen & Engell, 2020], which serves as a baseline for the tests conducted and presented in the report.
The best performing iteration of the CVAE was trained for 800 epochs, with a warm-up effect during the first 50% of training, using ReLU activation functions and a learning rate of 2e-4. This CVAE was able to create a cluster which contained roughly 1/3 of the noise present in the ground-truth dataset with 87% certainty. The remaining clusters are poorly resolved, with certainties in the range of 25%-50%. This indicates a CVAE which is best suited for categorizing noise specifically. The CVAE is able to reconstruct normal lightning events well, which are also the most well defined event in the dataset. Additionally, it performs poorly when reconstructing noisy data, or false triggers, which is the expected result, given that these are ideally underrepresented in the dataset. For future work, looking into the architecture of the CVAE could be an obvious next step, or more rigorous tests with regards to hyperparameters than has been done. This, however, requires more computational power, or time.
A git-repository of the implemented CVAE and the tests can be found on `https://github.com/RCJoergensen/Synthesis_CVAE_ASIM`

Contents

1 Introduction

The Modular Multispectral Imaging Array (MMIA) is one of two instruments onboard the Atmosphere Space Interaction Monitor (ASIM) onboard the International Space Station (ISS). The MMIA instrument measures emission lines of Transient Luminous Events (TLEs) on top of thunderclouds in three different wavelengths; 337nm, 180-230nm (UV), and 777.4nm [Chanrion *et al.*, 2019]. TLEs cover a range of different events such as Blue Jets, Sprites, and Elves which we are just now beginning to get a better understanding of [Neubert *et al.*, 2020, 2021]. However at present time, the process of simply reviewing the data from MMIA is a tedious process, which requires either manually examining the data files [Neubert & Chanrion, 2020], or developing specific algorithms for finding selected types of events [Jørgensen & Olesen, 2020]. The downside of developing event specific algorithms lies in the necessity of first manually finding an example of these events, which the algorithm can be based on. This leads to potentially missing interesting events, and does not solve the underlying manual problem. Instead, a more robust approach could potentially greatly remove the burden of manually examining the data when initially reviewing it. With this in mind, the intention is to build a neural network that can help with the process of selecting interesting datafiles in the ASIM dataset, or at the very least filter noise and Cosmic Rays out, which are seen as false triggers, such that only data of interest remains.

From previous studies, it is apparent that the events of interest are often highly uncorrelated in the three measured wavelengths [Neubert *et al.*, 2021, 2020; Jørgensen & Olesen, 2020], whereas normal lightning events, that also trigger the instrument, are often highly correlated. Therefore, an early attempt at building a neural network for TLE detection has utilized Convolutional Variational Auto-Encoders (CVAE) [Jørgensen & Engell, 2020]. By analyzing the latent space of the CVAE the goal is to be able to detect outliers which are uncorrelated in the three measured wavelengths, corresponding to the types of events of interest. The CVAE utilized concatenating the three measured wavelengths, before running them through four convolutional layers, max pooling, and batch normalization, before finally using three fully connected layers. The CVAE used a beta-distribution as its observational model, and normalization of the three input wavelengths with respect to the MMIA instruments saturation points [Jørgensen & Engell, 2020]. This preliminary attempt was able to construct the over-all features and intensity of the strong narrow-band wavelengths 337nm and 777.4nm, but was not able to accurately model the much weaker broadband 180-230nm. Additionally, an issue with the boundary of the reconstruction appears present in some reconstructions [Jørgensen & Engell, 2020].

The CVAE presented in [Jørgensen & Engell, 2020] will be the basis of the continued work in this project. The goal will be to optimize the CVAE, in order to differentiate between different kind of triggers in the dataset. This will include finding a proper way to score the tests of the network, in order to validate which configuration yields the best overall results.

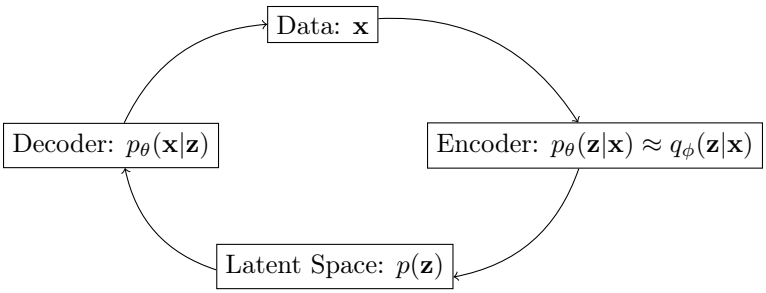
First the relevant theory behind the CVAE will be presented in section 2, along with the concepts that are being tested and used, such as warm-up effect and batch normalization. In section 3, the implementation itself will be explored; how the data is prepared for the CVAE, how the training and validation set has been selected, the overall architecture of the CVAE, and finally how, and which tests, are planned. In section 4 the results are presented. This is only the raw results from the tests, and some examples from the best performing iteration of the CVAE. The discussion of these results, and the limitations and problems with the CVAE and the way that it is being tested, can be found in section 5. Finally, a conclusion can be found in section 6.

2 Theory

In this section the relevant theory for the concepts used in the project will be covered. It is assumed that the reader has a general knowledge with regards to Neural Networks, their structure, learning algorithms, etc. Therefore, concepts such as Backpropagation and Stochastic Gradient Descent (SGC) will not explicitly be explained. Instead, the focus will be on the core concepts relevant to the implementet CVAE.

2.1 Variational Auto-Encoders (VAE)

The idea behind the VAE is to have two neural networks working together; the Encoder and the Decoder. The very general idea is that the Encoder takes input data ($\mathbf{x}$), and encodes it over the latent space ($\mathbf{z}$). When sampling from the latent space, the Decoder can then reconstruct the original data from the latent space. In order to do this, it is necessary that the Encoder learns the posterior distribution of the data, $p_\theta(\mathbf{z}|\mathbf{x})$, also called the *inference model*. Likewise, the Decoder has to learn the *generative model*, $p_\theta(\mathbf{x}|\mathbf{z})$ [Kingma & Welling, 2019, Chap. 2]. This allows the generative model, the Decoder, to accurately reconstruct the data from the latent space, while the Encoder accurately encodes the data to the latent space, in a way that the generative model 'understands'. This can be visualized as the following scheme:



However, this is not a trivial task. In the scheme above, a new approximate distribution, $q_\phi(\mathbf{z}|\mathbf{x})$, is introduced in the Encoder term. Therefore, let us start by looking into the Encoder, and why this approximation is necessary. It is required to find a way to compute and learn the posterior model, $p_\theta(\mathbf{z}|\mathbf{x})$ for the Encoder. From Baysian statistics, Bayes rule tells us that we can update probabilities as new knowledge is acquired as:

$$p_\theta\left(\mathbf{z}\left|\mathbf{x}\right.\right) = \frac{p_\theta\left(\mathbf{x},\mathbf{z}\right)}{p_\theta\left(\mathbf{x}\right)} \tag{1}$$

However, this is intractable, since it requires us to be able to evalute $p_\theta(\mathbf{x})$, which is given as [Memarzadeh *et al.*, 2020; Sønderby *et al.*, 2016]:

$$p_\theta\left(\mathbf{x}\right) = \int_{\mathbf{z}} p_\theta\left(\mathbf{z},\mathbf{x}\right) d\mathbf{z} \tag{2}$$

In order to get around this problem, the *approximate posterior*, $q_\phi(\mathbf{z}|\mathbf{x})$, is introduced which utilizes *Variational Inference* (VI) [Memarzadeh *et al.*, 2020]. A common choice for $q_\phi(\mathbf{z}|\mathbf{x})$ is the Gaussian family, for instance the isotropic Gaussian:

$$q_\phi\left(\mathbf{z}\left|\mathbf{x}\right.\right) = \mathcal{N}\left(\mathbf{z}\left|\mu_\phi\left(\mathbf{x}\right),\sigma_\phi\left(\mathbf{x}\right)\mathbf{I}\right.\right) \tag{3}$$

This lets us compute $p_\theta(\mathbf{x})$ by using the expectation over $q_\phi(\mathbf{z}|\mathbf{x})$:

$$p_\theta\left(\mathbf{x}\right) = \int_{\mathbf{z}} p_\theta\left(\mathbf{x},\mathbf{z}\right) d\mathbf{z} = \int_{\mathbf{z}} \frac{q_\theta\left(\mathbf{z}\left|\mathbf{x}\right.\right)}{q_\theta\left(\mathbf{z}\left|\mathbf{x}\right.\right)} p_\theta\left(\mathbf{x},\mathbf{z}\right) d\mathbf{z} = E_{q_\theta(\mathbf{z}|\mathbf{x})}\left[\frac{p_\theta\left(\mathbf{x},\mathbf{z}\right)}{q_\phi\left(\mathbf{z}\left|\mathbf{x}\right.\right)}\right] \tag{4}$$

Now it is possible to determine a training criterion $\mathcal{L}$. This is achieved by computing the Evidence Lower Bound (ELBO), which comes from Jensen's Inequality and the Kullback-Leibler Divergence (KL). In broad terms the KL tells us about the difference between two distributions. This leads to the training criterion [Sønderby *et al.*, 2016; Kingma & Welling, 2019]:

$$\begin{aligned}\log p\left(\mathbf{x}\right) &\geq E_{q_\phi(z|x)}\left[\log \frac{p_\theta\left(\mathbf{x},\mathbf{z}\right)}{q_\theta\left(\mathbf{z},\mathbf{x}\right)}\right] = \mathcal{L}\left(\theta,\phi;\mathbf{x}\right) \\ &= -KL\left(q_\phi\left(\mathbf{z}\,|\mathbf{x}\right)\|p_\theta\left(\mathbf{z}\right)\right) + E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[p_\theta\left(\mathbf{x}\,|\mathbf{z}\right)\right]\end{aligned} \tag{5}$$

The remaining problem now is how to deal with the Expectation operator in eq. (5), since it requires us to sample $\mathbf{z} \sim \mathbf{q}_\theta\left(\mathbf{z}\,|\mathbf{x}\right)$, which is a stochastic process and cannot be backpropagated. To solve this, the *Reparameterization Trick* is used. This introduces a noise variable ε, with a fixed distribution $p(\varepsilon)$. Additionally a deterministic and differentiable transformation $g_\phi\left(\varepsilon,\mathbf{x}\right)$ is also introduced [Kingma & Welling, 2019, Chap. 2.4]. It is now possible to express the sampling as:

$$\mathbf{z} \sim q_\phi\left(\mathbf{z}\,|\mathbf{x}\right) \;\equiv\; \mathbf{z} = g_\phi\left(\varepsilon,\mathbf{x}\right), \varepsilon \sim p\left(\varepsilon\right) \tag{6}$$

As shown in [Kingma & Welling, 2019, Chap. 2.4] this makes it possible to compute the gradient as:

$$\nabla_\phi \mathcal{L}\left(\mathbf{x}\right) = \nabla_\phi E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[f_{\theta,\phi}\left(\mathbf{z},\mathbf{x}\right)\right] = E_{p(\varepsilon)}\left[\nabla_\phi f_{\theta,\phi}\left(\tilde{\mathbf{z}},\mathbf{x}\right)\right] \tag{7}$$

Where: $\tilde{\mathbf{z}} = g_\phi\left(\varepsilon,\mathbf{x}\right), \varepsilon \sim p\left(\varepsilon\right), \; f_{\phi,\phi}\left(\mathbf{z},\mathbf{x}\right) = \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}$ and ε would typically be chosen as a Gaussian Distribution $\varepsilon \sim \mathcal{N}(0, I)$.

Next the latent space is also described by the prior distribution $p_\theta(\mathbf{z})$. This is usually chosen as a very simple distribution, e.g. a Gaussian distribution:

$$p(\mathbf{z}) := \mathcal{N}(0, I) \tag{8}$$

Finally the generative model in the Decoder is also described by a distribution $p_\theta(\mathbf{x}|\mathbf{z})$. This distribution depends more on the data that is being delt with. For instance, in binary pixel values, a Bernoulli distribution would make sense, since the expectation is binary outputs. However, for the purpose of this project, a Beta-distribution was chosen:

$$p_\theta(\mathbf{x}|\mathbf{z}) := \text{Beta}(\alpha, \beta) \tag{9}$$

The Beta-Distribution is a continous probability distribution defined in the interval [0,1] by the two shape paremeters α and β. The probability density function (PDF) is given as:

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B\left(\alpha,\beta\right)} \tag{10}$$

Where $B\left(\alpha,\beta\right) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$ and Γ is the Gamma Function: $\Gamma\left(n\right) = (n-1)!$ [NIST, 2003].

2.1.1 Convolutional Variational Auto-Encoders

The Convolutional Variational Auto-Encoder (CVAE) is simply an extention of the normal VAE, which uses one or more convolutional layers. Any convolutional network is simply a network which uses convolutions, instead of general matrix multiplication, in at least one of its layers [Goodfellow *et al.*, 2016, Chap. 9]. The 1-dimensional

convolution refers to the mathematical operation, defined in discrete time as [Goodfellow *et al.*, 2016, Chap. 9, eq. 9.3]:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)\, w(t-a) \tag{11}$$

Convolutional layers are also very useful in, for instance, 2-dimensional datasets like images and image classification tasks. However, these will not be covered here, as the focus of this report is on 1-dimensional time series. Additionally, the convolutional layers usually take advantage of *pooling*. Pooling refers to the act of grouping together output parameters from a layer, and instead of passing the true output parameters to the next layer, a summary statistic is passed on. Many different kind of pooling exists, but the one used in this report is *max pooling*. Max pooling is grouping together parameters, and only returning the parameter with the maximum value [Goodfellow *et al.*, 2016, Chap. 9].

One of the advantages of CVAEs is the ability to detect anomalies in high dimensional time-series. The convolutional layers in the CVAE is able to detect features in the high dimensional data, concatenate them, and map them to the latent space. As demonstrated by [Memarzadeh *et al.*, 2020], if the training dataset consists of an unbalanced ratio of nominal to anomalous data, it is possible to learn an optimal mapping for nominal data, but not for anomalous data. This results in a low reconstruction error when dealing with nominal data, but a high reconstruction error with anomalous data. By utilizing this, [Memarzadeh *et al.*, 2020] have shown that they are able to significantly outperform all their baseline models for anomaly detection in high-dimensional heterogeneous time series.

2.1.2 β-VAE & Warm-up

From the training criterion of the VAE, eq. (5), it is apparent that the training consists of an equilibrium between the reconstruction term, $E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[p_\theta(\mathbf{x}|\mathbf{z})\right]$, and the regularization term, $KL\left(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})\right)$. The goal of the regularization term can be seen as trying to prune away irrelevant latent units. However, if the regularization term is too strong in the early stages of training, this can lead to latent units being removed before a useful representation is learned, reducing the capacity of the overall VAE [Sønderby *et al.*, 2016]. In order to combat this, the β-parameter is introduced. The resulting learning criteria then becomes [Sønderby *et al.*, 2016]:

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = -\beta KL\left(q_\theta(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})\right) + E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[p_\theta(\mathbf{x}|\mathbf{z})\right] \tag{12}$$

This can effectively be seen as a way of penalizing the regularization term, to allow the reconstruction term to have a higher influence on training. This is analogous to the regularization parameter used in e.g. Tikhonov Regularization.
However, unlike in Tikhonov Regularization, where the regularization parameter is often kept constant, it can be beneficial to vary the value of β during training, as shown in [Sønderby *et al.*, 2016]. By increasing β lineraly from 0 to 1 during training, it allows the VAE to learn a useful representation of the latent units, before the regularization kicks in. This reduces the chances of a latent unit being pruned away before the representation is learned, and is refered to as Warm-up. [Sønderby *et al.*, 2016] shows that this allows more latent units to remain active, especially in VAE with deeper network structure.

2.2 Activation Functions

A fundamental part of the structure of neural networks are the activation functions connecting layers in the network. It is necessary that the activation functions are non-linear to ensure that each layer is indepdent [Ygor, 2020; Goodfellow *et al.*, 2016, Chap. 6]. If two layers in a neural net are not independent, these would

mathematically be equivallent to a single layer, defeating the purpose of multi-layer neural network [Ygor, 2020; Goodfellow *et al.*, 2016, Chap. 6]. The choice of choosing an activation function is not straight forward, and a heuristic approach is necessary when making the decision [Szandała, 2020]. Additionally, a wide variety of activation functions exists, but only two will be discussed here, as these are the relevant ones for this project.

2.2.1 Rectified Linear Unit (ReLU)

The Rectified Linear Unit (ReLU) is one of the most common activation functions [Szandała, 2020 ; Goodfellow *et al.*, 2016, Chap. 6]. One of the reasons being that it is mathematically simple, making it fast to use in computation, while yielding good results [Szandała, 2020]. The ReLU activation function is a piecewise function, defined as:

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x \text{ if } x > 0 \\ 0 \text{ if } x \leq 0 \end{cases} \tag{13}$$

Because the function returns 0 when the input is 0 or less, this is exactly what causes the function to be computationally inexpensive, and resulting in sparse networks [Leung, 2021]. This also solves the problems with vanishing gradients, that other activation functions may experience [Leung, 2021; Szandała, 2020]. Additionally, it has no upper cap, like some other activation function has, e.g. the sigmoid function, which has not been discussed here. However, there are also problems with the ReLU function, for instance the *dying ReLU* issue. This happens when the input becomes too negative, and the ReLU functions get stuck returning 0, preventing the network from learning [Leung, 2021]. This issue can for instance happen if the learning rate is too high. When updating the weights, the new weights are computed as:

$$w_k^* = w_k - \eta \frac{\partial C}{\partial w_k} \tag{14}$$

Where w_k^* is the new updated weight, w_k is the old weight, η is the learning rate, and C is the cost function [Nielsen, 2015, Chap. 1, eq. 16]. If the learning rate, η, is too large, then $\eta \frac{\partial C}{\partial w_k} > w_k$ which leads to the updated weight being negative, and the ReLU function returning 0. This can result in the *dying ReLU* problem, which ends up making learning impossible. One possible solution is setting a lower learning rate, but the *Leaky Rectified Linear Unit* function can also be considered as an alternative.

2.2.2 Leaky Rectified Linear Unit (Leaky ReLU)

The Leaky Rectified Linear Unit (Leaky ReLU) is an extension of the ReLU activation function. It aims to solve the problem of the *dying relu*, by enabling outputs below 0. The function is in many ways similar to the ReLU function, eq. (13), and is defined as the piecewise function:

$$\text{Leaky ReLU}(x) = \max(0, x) + \alpha \cdot \min(0, x) = \begin{cases} x \text{ if } x > 0 \\ \alpha \cdot x \text{ if } x \leq 0 \end{cases} \tag{15}$$

Where α is a slope parameter, usually with a low value; $\alpha << 1$ [Szandała, 2020]. This prevents the problem of dead, or inactive, nodes that get stuck in 0 values, thereby enabling learning to continue. It also reduces the problem of high learning rates resulting in the *dying relu* problem, since a negative value no longer returns a 0 value. For this reason, if the *dying relu* is suspected to be present, the Leaky ReLU function is a good next step in testing. This intuitive reasoning is in agreement with [Szandała, 2020], who proposes ReLU as the go-to first choice when testing activation functions, and the Leaky ReLU function as the next try *if* the *dying relu* problem is suspected.

2.3 Batch Normalization

When training deep neural networks, the input to each layer changes due to the updating of weights and activation functions. However, this can slow down the process of training, requiring low learning rates to achieve stable learning in the network. This problem of changing inputs between layers, are refered to as *Internal Covariate Shift* by [Ioffe & Szegedy, 2015], who proposes Batch Normalization as a solution to this problem. The general idea is to fix the distribution of layer inputs as the training progresses, and incorporates it as a part of the network structure itself, as opposed to simply fixing the initial input. For a layer with d-dimensional input $x = (x_1, ..., x_d)$, the normalization is defined as:

$$\hat{x}_k = \frac{x_k - E\left[x_k\right]}{\sqrt{Var\left[x_k\right]}} \tag{16}$$

Where x_k is the k'th input to the layer, E is the expectation operator, Var is the variance operator, and $\hat{x}_k$ is the new normalized input [Ioffe & Szegedy, 2015]. Additionally, the two scale parameters γ_k and β_k are introduced, used to scale and shift the normalized value, to ensure that the layers ability to represent the input transformation remains unchanged. These are introduced as:

$$y_k = \gamma_k \hat{x}_k + \beta_k \tag{17}$$

Where y_k is the final output from the batch normalization. The implementation proposed by [Ioffe & Szegedy, 2015] on a batch, B, of size m, is straight forward, and follows the 4 steps:

1. Compute Batch Mean: $\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
2. Compte batch Variance: $\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}\left(x_i - \mu_B\right)^2$
3. Normalize Batch: $\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$
4. Scale and shift: $y_i \leftarrow \gamma \hat{x}_i + \beta$

A new parameter, ε, is introduced in step (3) here, which is simply a constant added for numerical stability. [Ioffe & Szegedy, 2015] concludes that the introduction of Batch Normalization significantly increases the rate of training in neural nets, allowing higher learning rates and less careful initialization, while achieving the same, or better, performance as earlier state-of-the-art networks they compare to.

3 Method

In this section the reasoning behind implementation choices will be explored, and what kind of data is being dealt with. The focus will be on what kind of data the ASIM instrument produces, and from this, how the training and validation datasets have been selected. Additionally, a focus will be on why the CVAE is proposed as the network architecture, which kinds of tests are planned for the CVAE, and how each iteration of the CVAE is scored, enabling us to determine how well a set a test parameters fares.

3.1 Data

As discussed in the Introduction, the data consists of observations from the MMIA instrument, from which the goal is to find TLEs ontop of thunderclouds. An overview of the different kind of TLEs can be found on fig. 1. The data consists of measurements of the emission lines in three different wavelenghts; 337nm (Nitrogen second Positive N_2P2), 180-230nm (UV), and 777.4nm (Atmospheric Oxygen, O). These are discrete time-series, with a temporal resolution of 10 µs [Chanrion *et al.*, 2019, Table 2]. Each time series is a couple hundred ms long, depending on the observed event. An example of one of the time series can be seen on fig. 2. The instrument has two data products, Level 0 and Level 1. Level 0 is the raw photon count, whereas the Level 1 has been converted to the energy of the photon flux, for physical interpretation. The focus will be on Level 1, but the shape of Level 0 and Level 1 does not differ, as the conversion between photon count and photon flux conservs the overall shape [Chanrion *et al.*, 2019]; Level 1 is simply easier to physically interpret.
From fig. 2 a few things are worth mentioning with regards to the ratio between the different spectra; the UV spectra is generally much weaker than the 337nm and 777.4nm, typically on the order of 2-3 magnitudes. Additionally, the peaks observed in the spectra can vary a lot; on fig. 2 the first peak shows an *almost* delta like structure, like those discussed in [Neubert *et al.*, 2021]. True delta peaks are however also present in the dataset, as discussed in [Jørgensen & Olesen, 2020], which typically correspond to interference from Cosmic Rays; these are generally undesired. Peaks with slower build ups are also very common, for instance the peaks seen around the 500-600ms mark on fig. 2. These typically correspond to normal lightning deeper in clouds, and make up a large majority of the data from MMIA.

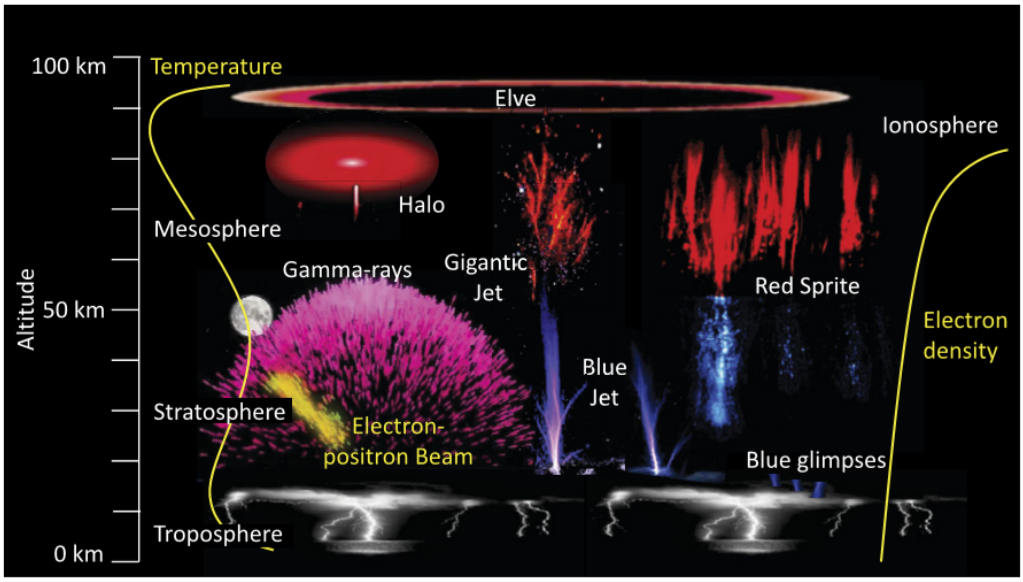


Figure 1: An overview of the different kind of known TLEs [ESA, 2018]

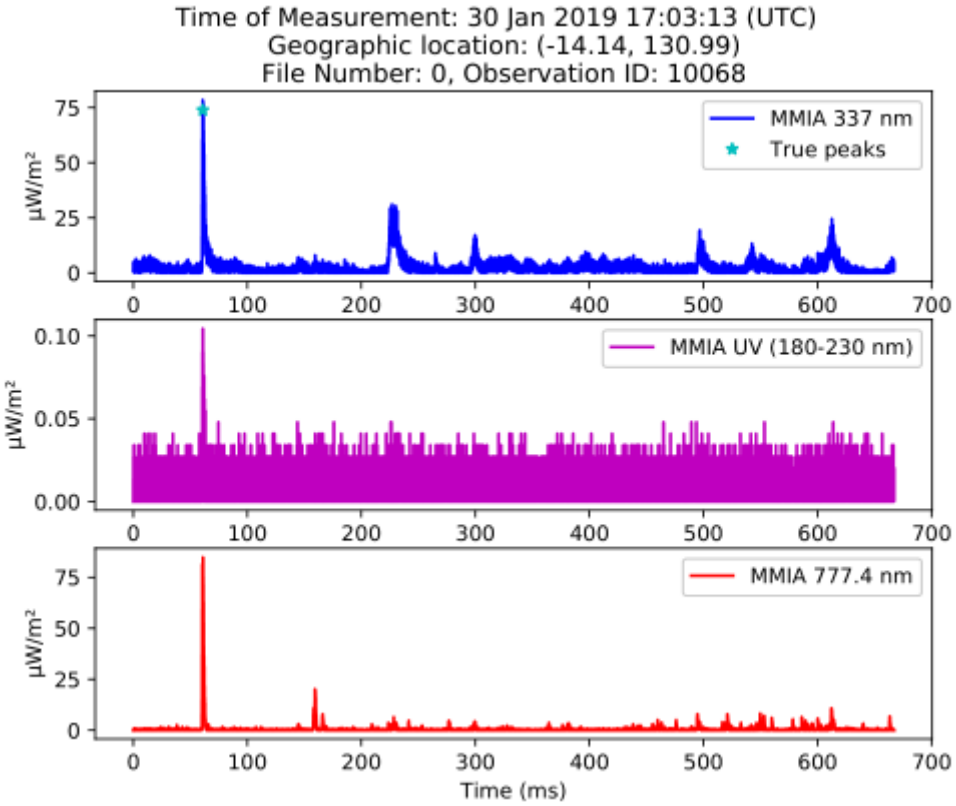


Figure 2: Example of a time series from the MMIA instrument on ASIM, showing Level 1 data [Jørgensen & Olesen, 2020].

3.2 Training and Validation datasets

Selecting the training and validation datasets are an important part of being able to properly train any neural net. Generally, the goal is to get a randomly sampled dataset, to avoid any bias being present during training. Since the ASIM instrument is on board the International Space Station (ISS), there are multiple periodic influences that should be considered; the orbit of ISS itself (∼90 Minutes, ∼16 orbits in 24h), but also both seasonal and geographic (latitudinal, longitudinal) influences. For instance, it is suspected that delta-like peaks in 337nm (Blue Jets) are biased towards the hotter equator, as early investigation seem to agree with [Jørgensen & Olesen, 2020]. However, this is still unconfirmed, and just a suspicion [Neubert & Chanrion, 2020]. In order to get around these biases, the dataset has been picked as follows: A full days data (00:00 - 23:59, ∼16 orbits) has been selected every 7th day, for the entire year 2019 (1. Jan - 31. Dec). This results in a total of 42580 datafiles. These has been split randomly 90% training, and 10% validation. However, since training has been conducted on a home computer, only 30% of this dataset has been used, each time selecting a random 30% from both training and validation. This results in a final training dataset of 11496 datafiles, and a validation dataset of 1277 datafiles.

3.3 CVAE Architecture

Since the data consists of time series of three different spectra, it is necessary to relate the three spectra to each other somehow. As seen from fig. 2, and [Neubert *et al.*, 2021], both correlated peaks and uncorrelated peaks are present in the data. For this reason, it was chosen to concatenate all three spectra together for the input to the

CVAE, and use a series of 1D convolutional layers to detect any features. This is the same process as described by [Memarzadeh *et al.*, 2020], who also deal with 1D time series, and whishes to determine anomalies. However, [Memarzadeh *et al.*, 2020] wishes to detect and remove all anomalous data, since it is not useful. Unfortunetly, it is not as straight forward with the ASIM dataset. Since the majority of the dataset is expected to be normal lightning events, both Cosmic Rays and e.g. Blue Jets are seen as 'anomalous' compared to normal lightning, but Blue Jets are desired, and Cosmic Rays are not. These are just two out of a long variety of events, which can have different and yet unknown features. Therefore, it is not desireable to simply determine and filter all anomalous datasets out. Rather, a way to distinguish between the different types of anomalous data would be ideal. Therefore, the hope is that a CVAE will be able to accurately distinguish between the different kind of 'anomalous' datasets.

The entire architecture of the final implemented CVAE can be seen on fig. 3. The over all structure is heavily inspired by [Memarzadeh *et al.*, 2020], since they had already demonstrated a working CVAE for anomaly detection in 1D time-series. However, it was chosen to include Batch Normalization in in the convolutional layers, with the intentions of increasing stability in training, and hopefully allow higher learning rates. Additionally, as discussed by [Ioffe & Szegedy, 2015], batch normalization can in some cases work as a regularization, making dropout redundant. For this reason, dropout has not been included, as Batch Normalization was chosen in favor of it.

The Encoder uses max pooling inbetween each layer of the convolutional layers. To reverse this in the Decoder, it was chosen to use upsampling. This works for the most part, however, the last upsampling is hardcoded to upsample to n = 14997, instead of using the usual multipler of 2. This is because simple upsampling by a factor of two made it impossible to ever hit the number 14997, which was the expected dimensions of the layer, in order to match the Encoder.

Taking a look at the input data, it was chosen to both slice up the time-series and normalize them before inputting them. The slicing was for computational reasons. The original time-series have a varying length of 20.000 to 70.000 samples in each spectrum, resulting in the concatenated time-series having dimensions between 1x60.000 - 1x210.000. Additionally, the CVAE expect to have a fixed input length, which also was not the case. Therefore, slicing the time-series every 5000th sample solves both these problems; the time-series are significantly shorter making them less computationally heavy and fixes the concatenated spectrum to 1x15.000 samples. Additionally, it was chosen to normalize the data in order to add stability to training. Since it was already chosen to normalize the data, it followed naturally to choose a Beta distribution for the generative model. Since the Beta distribution in Pytorch is defined in the]0;1[interval, simply clamping the final output layer to [0.000001, 0.99999] enables the output to be directly used in the Beta distribtuion for the generative model. One thing to be considered of though, is that it requires both the α and β parameter. Therefore, to match the concatenated spectrum of 1x15.000 samples, an additional convolutional layer, having the same function as upsampling, was added to ensure the final output from the Decoder had dimension 1x30.000. By chunking this into 2x15.000 both the α and β parameter is provided.

Finally, the training criteria, the Loss function, as described by eq. (12), becomes:

$$\mathcal{L}(x) = \frac{1}{L}\sum_{i=1}^{L} \log p_\theta(\mathbf{x}|\mathbf{z}_i) - \beta \cdot \frac{1}{2N}\sum_{i=1}^{N}\left(\sigma_i^2 + \mu_i^2 - 2\log(\sigma_i) - 1\right) \tag{18}$$

Where σ and μ^2 refer to the mean and standard deviation of the inference model $q_\phi(\mathbf{z}|\mathbf{x})$. This is in agreement with the expected training criteria, where the first terms describes the expectation value of the Decoder, and the second term decribes the KL, with the added β effect.

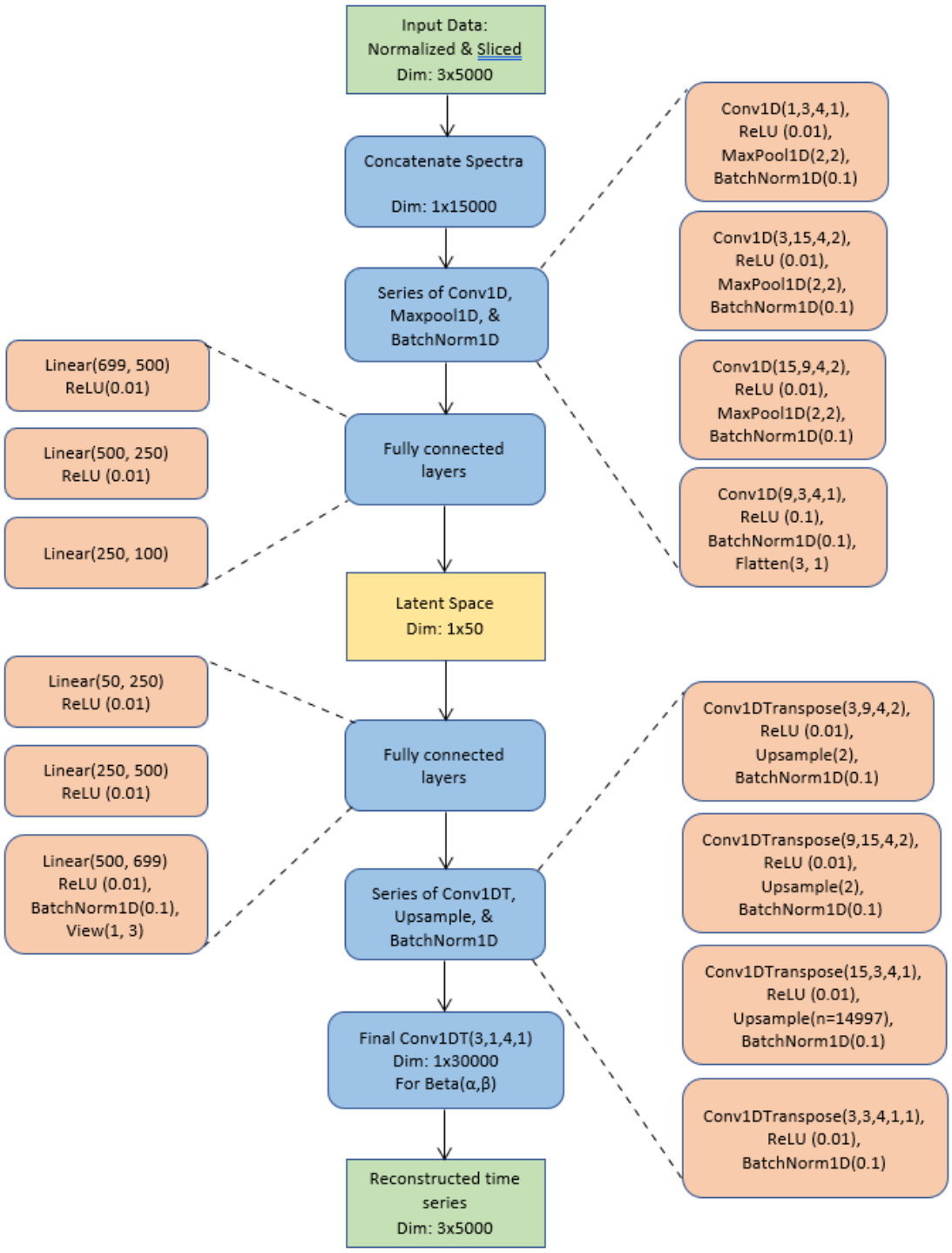


Figure 3: Architecture of the final implemented CVAE. The input refers to; Dim: Output Dimension of layer; Conv1D(in_channels, out_channels, kernal_size, stride, optional: output_padding); Relu(); MaxPool1D(Kernal_size, Stride); BatchNorm1D(momentum); Linear(Input_Size, Output_Size); Upsample(multiplier *or* n = Fixed Number); Flatten(in_channel, out_channel); View(in_channel, out_channel).

3.4 Testing the CVAE

Since the project is an extension of the earlier version presented in [Jørgensen & Engell, 2020], the focus will be on testing parameters that the time limitations did not allow in the original project. The original CVAE will be used as a baseline for the tests, to have something tangible to compare with.
First off, the earlier version never implemented the β-CVAE. [Sønderby *et al.*, 2016] demonstrated that the β-CVAE with warm-up (WU) was effective in keeping latent units active in deeper neural networks. Therefore, testing how the CVAE improves with added WU is of high priority. WU periods in the range of 25% to 50% of training will be considered. Additionally, the original CVAE was trained for a maximum of 300 epochs due to time limitations; longer training periods will be tested this time of 500 and 800 epochs.
In the original CVAE an unintentional Dropout layer was also present between the Convolutional and Linear layers. This will be fixed and replaced with a Batch Normalization layer to remain consistent with the rest of the CVAE; [Ioffe & Szegedy, 2015] also mentions that Batch Normalization can in places replace Dropout.
The baseline CVAE never tested changes in the learning rate. The learning rate for the baseline CVAE was set at 1e-4; tests of half that (5e-5), and double (2e-4), will be tested, to see if it improves results. Since the CVAE utilizes the ReLU activation function, increasing the learning rate can be expected to lead to stability problems in training, due to the *dying relu* problem. Therefore, changing to the LeakyReLU activation function will also be tested, with the goal of achieving more stable and faster training.

3.5 Scoring the CVAE

When testing the CVAE it is necessary to have a way of evaluating how well a given test does. In order to do this, some way of scoring the CVAE is needed. One of the big obstacles of the ASIM dataset is, that it is a new dataset that is still being investigated, which means that a fully classified datasets does not exist. The dataset is continiously being examined, and new physical phenomena are still being found, e.g. [Neubert *et al.*, 2020], [Neubert *et al.*, 2021]. However, this means that a large, definitiv ground-truth dataset is impossible to use when scoring the CVAE. To get around this problem, a new ground-truth dataset has been made, based on the results found in the original CVAE from [Jørgensen & Engell, 2020]. The procedure has been to use a K-Means clustering algorithm on the latent space of the original CVAE, and determine how many well defined clusters could be made. This turned out to be roughly 5 clusters. By inspecting each clusters, a classification was given to the events found in the given cluster, based on their shared features. These have turned out be very broad classes, and are as follows;

1. HCHA: High Correlation, High Activity
2. HCLA: High Correlation, Low Activity
3. LCHA: Low Correlation, High Activity
4. LCLA: Low Correlation, Low Activity
5. N: Noise

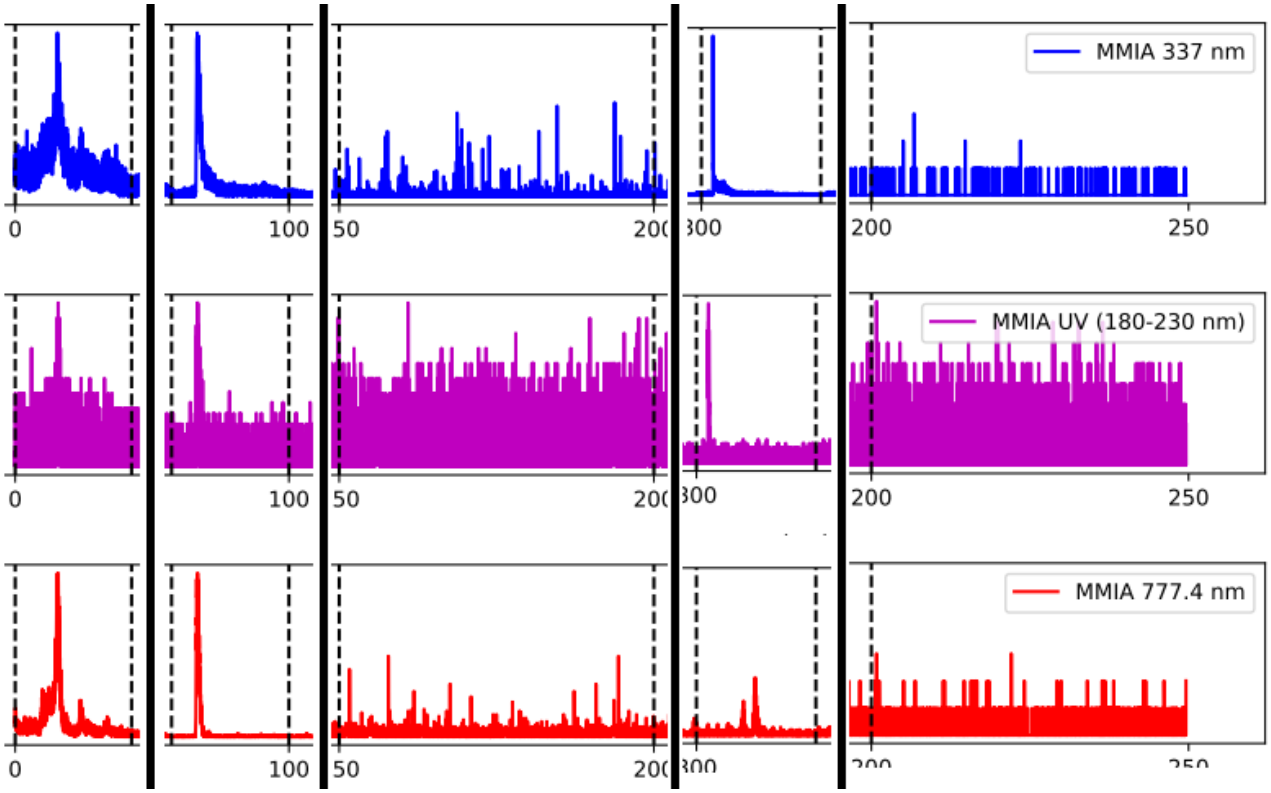


Figure 4: The 5 classes used for the ground truth set, seperated by the vertical black line. These are samples from different events to display all classes, which is why the time-axis is off. From left to right the classes are; HCHA, HCLA, LCHA, LCLA, N.

An example of each class be seen on fig. 4, where a frame, or slice of 5000 samples as given to the CVAE, is seen. Each represents one of the classes. These are just examples, and can vary quite a lot, but these are some of the clearner examples;
HCHA has highly correlated features in all 3 spectra, but also a lot of noise, or "activity", lots of small peaks indicating activity, but still an underlying highly corelated signal. This is probably indicitative of very strong lightning over very active lightning clouds.
HCLA is also highly correlated, but much cleaner, and not as many small peaks; thereby 'low acitivity'. This could indicate lightning clouds with fewer overall lightning discharges, but often of much stronger amplitude.
LCHA has a lot of random peaks, but no apparent underlying correlation. This class is hard to physically interpret, as it does not appear to agree with what we generally expect to see [Neubert & Chanrion, 2020].
LCLA is uncorrelated, especially between 337nm (blue) and 777.4nm (red), where we generally expect correlation in normal lightning. These are quite interesting, and the example shown on fig. 4 is almost a spot on replication of the event presented in [Neubert *et al.*, 2021].
N is similar to LCHA, in the fact that no structure is seen. However, the signal is much, much weaker than LCHA, indicating white noise, or possibly Cosmic Rays triggering the instrument.
It is important to note that these are by no means definitiv classes, and a more thorough investigation is needed to get a true physical interpretation of the classes. However, given the scope of the project, this is the baseline chosen when scoring the CVAE. 50 randomly sampled events have been chosen for the baseline, each divided up into slices of 5000 samples, as the CVAE inteprets it, and given a label. This gives a total of roughly 500 labelled events in the ground-truth dataset. The classification was done with the help of two fellow student from

DTU Space, who have worked extensively with the ASIM dataset; Emilie P.P.W. Olesen & Emma A. Nielsen. When scoring the CVAE, after training is complete, the CVAE is run on all 500 events in the ground-truth dataset. A K-Means clustering is run on the latent space, with the expected 5 clusters. The number of each class in a given cluster is counted up, and the most represented class in a given cluster is assigned to the cluster, with a percentage indicating how many of the events in the cluster the label represents. For instance, if cluster 0 had 50 events, and the 50 events where distributed as; 5 HCHA, 7 HCLA, 2 LCHA, 1 LCLA, 35 N, then cluster 0 would be given the class "N" with the percentage $\frac{35}{50} \cdot 100 = 70\%$. The idea being, that the optimal version of the CVAE will represent each class in one of its 5 clusters, with a high degree of certainty. The visualization of the latent space is done in 2D space, using t-SNE for dimensionality reduction from the 50D latent space. Finally, the mean distance to the cluster center, of each cluster, is computed. This is not done with any certainty that it is a good parameter to look at, more an intuitive test, and it is fairly straight forward to do. The idea being, that a 'well-defined' cluster would intuitively be evenly spread close to the center of the cluster, giving it a low mean distance to cluster center.

4 Results

With the ground-truth dataset in place, it became possible to quantify how well an iteration of the CVAE performes. In order to have a baseline for the tests, the ground-truth dataset was first run through the old CVAE presented in [Jørgensen & Engell, 2020], which used no Warm-up effect. The classification of each cluster is computed, along with the percentage the classification is based on, as described in section *3.5 - Scoring the CVAE*. Additionally, the Mean Distance to the Center of the Cluster (MDCC) is computed. The parameters used for each iteration is also listed in the first column, in order to keep track of which iteration of the CVAE is being delt with. This parameter column takes the form:
Epoch (Ep) [number of epochs], Warm-Up (WU): [Fraction of training with active warm-up effect], Activation-Function, Learning Rate (LR): [selected Learning Rate]
An overview of the results from the baseline and all the relevant tests performed are shown in table 1. A few specific numbers are highlighted in red. These are the best performing clusters in a given test; no red color is used if the test does not out-perform the earlier iteration of the CVAE.

Classification (%) **MDCC**	**Cluster 0**	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**
Baseline No WU	LCLA (39%) 7.16	HCHA (48%) 6.67	LCLA (32%) 6.53	N (35%) 6.07	LCLA (38%) 6.60
Ep 300, WU: 1/3 ReLU, LR: 1e-4	LCLA (39%) 7.80	LCLA (42%) 7.22	HCHA (45%) 7.23	**N (57%)** 5.88	HCLA/LCHA (24%) 5.89
Ep 500, WU: 2/5 ReLU, LR: 1e-4	LCLA (31%) 8.08	HCHA (31%) 10.60	LCLA (26%) 9.23	**N (66%)** 10.92	HCHA (40%) 9.39
Ep 800, WU: 1/4 ReLU, LR: 1e-4	LCLA (41%) 8.00	LCLA (25%) 6.96	LCLA (36%) 8.79	HCHA (50%) 7.95	N (42%) 7.40
Ep 800, WU: 1/2 ReLU, LR: 1e-4	LCLA (31%) 7.24	LCLA (42%) 8.53	LCHA (31%) 8.25	**N (74%)** 7.15	HCHA (43%) 9.39
Ep 800, WU: 1/2 ReLU, LR: 5e-5	N (36%) 6.04	LCLA (36%) 6.30	HCHA (53%) 8.82	LCLA (36%) 8.17	LCLA (41%) 7.44
Ep 800, WU: 1/2 ReLU, LR: 2e-4	**N (87%)** 9.29	N (52%) 13.04	HCLA (27%) 9.56	LCLA (34%) 10.78	HCLA (34%) 7.90
Ep 800, WU: 1/2 LeakyReLU, LR: 1e-4	N (68%) 10.88	LCLA (30%) 5.46	N (28%) 6.66	HCHA (48%) 7.34	LCLA (42%) 6.30
Ep 800, WU: 1/2 LeakyReLU, LR: 5e-5	HCHA (44%) 7.77	N (35%) 6.03	LCLA (42%) 7.85	LCLA (38%) 8.56	HCLA (36%) 8.87
Ep 800, WU: 1/2 LeakyReLU, LR: 1.5e-4	HCLA (35%) 8.89	HCLA (38%) 9.50	N (36%) 6.34	LCLA (48%) 7.98	HCHA (48%) 8.62

Table 1: An overview of the results from testing each iteration of the tested CVAE on the ground-truth dataset. The Classification and the percentage used for the classification is displayed, along with the Mean Distance to Cluster Center (MDCC). Red colors indicates the best performing cluster in a given test. If no red color is present, it indicates that the CVAE tested does not outperform the earlier iteration.

The CVAE with Epoch 800, warm-up periode of 50% of training, ReLU activaiton function and learning rate of 2e-4 had the cluster with the higest certainty; Cluster 0 consiting of 87% noise. Therefore, a closer look at the latent space and reconstruction of this specific CVAE will be presented here. However, all trained models can be found in the github repository (see Abstract), and a script for reproducing the above results aswell, along with the excel file of the groundtruth classification. However, as the ASIM dataset is not a publicly available dataset, raw data used for the ground-truth dataset is not to be found in the github repository.

On fig. 5 a feature map of the latent space, and a 2D projection of the latent space is shown. For the projection, the full latent space has first been clustered using K-Means, to determine 5 clusters. The classification of each cluster is then found, and the results are finally projected to 2D using the t-SNE algorithm. This shows 3 small clusters (0,1, and 4), and two large clusters, (2 and 3), which by far contains the majority of the encoded data. From the feature map, a few of the latent features start to show well-defined trends; feature 18 and 26 appear to generally have a very low value around -10, whereas feature 41 appear to generally have a large value around 10. The remaining features appear to be of middle values in the -5 to 5 range, but vertical features does seem to appear.

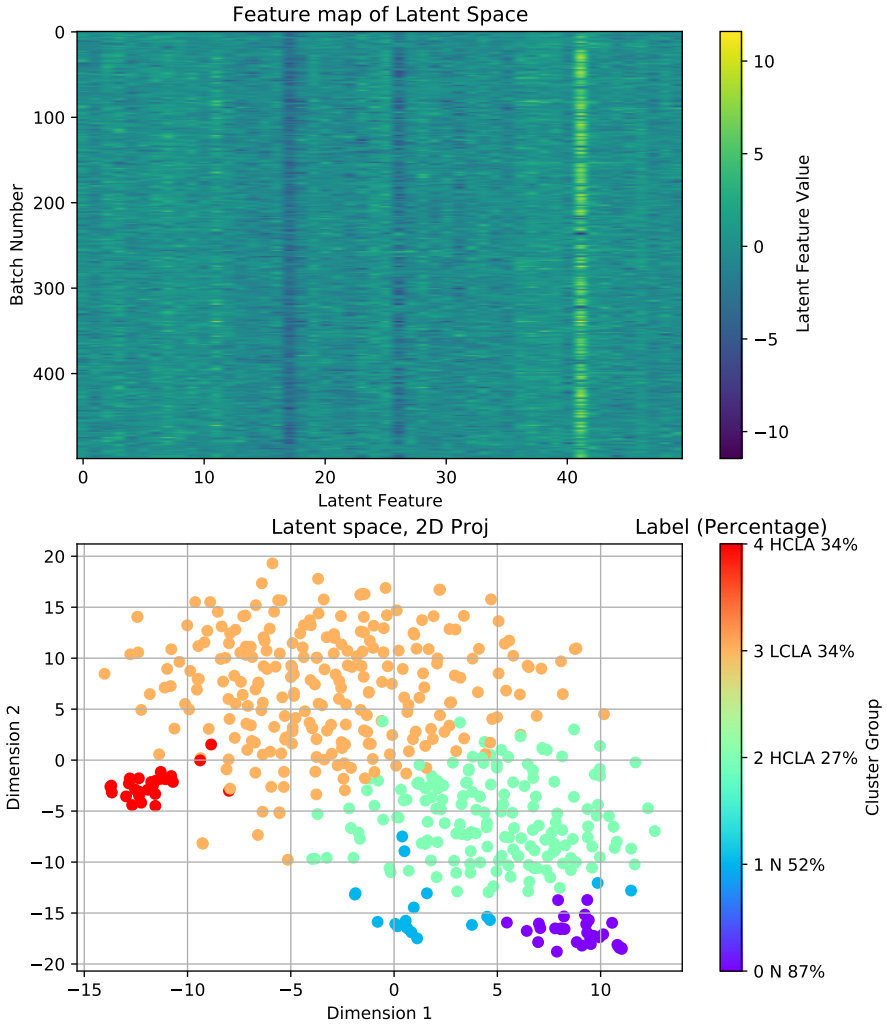
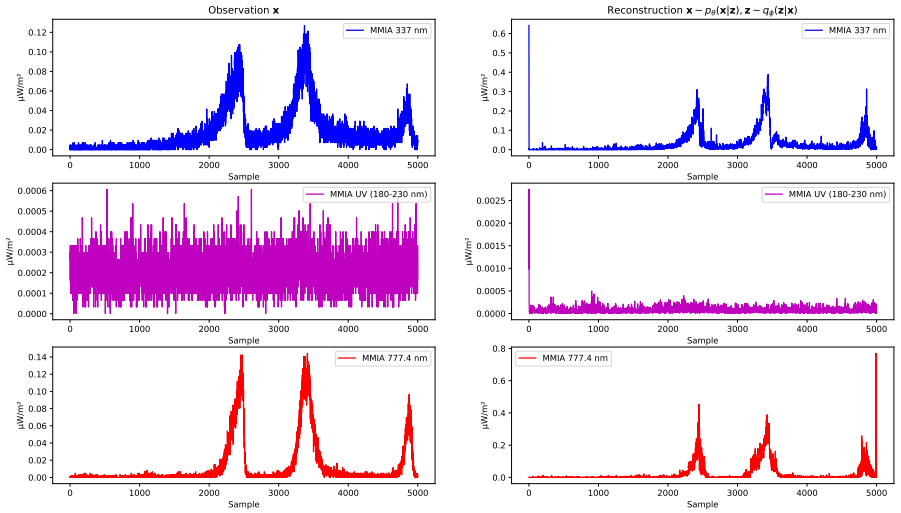
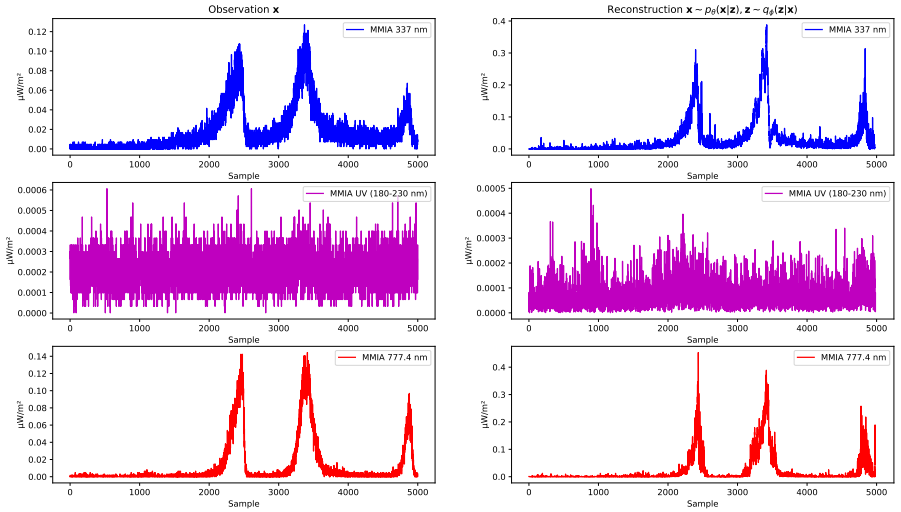


Figure 5: The latent space of the best performing CVAE iteration. The top showing a feature map of all latent, and on the bottom a 2D projections, using t-SNE, of the clustered latent space, from K-Means, along with the classification.

The baseline CVAE had problems with the boundaries of the reconstructions, as described in [Jørgensen & Engell, 2020]. As seen from fig. 6 the problem still persists; large spikes at the edge of the reconstruction are present. For this reason, when visually comparing the data and the reconstruction, the edges are omitted. This is *only* done when visually plotting the figures, to better compare features; for instance in the UV spectrum on fig. 6a it is hard to visually compare it to the original data. It is important to note that the data is in no way changed in the CVAE itself, this is only for visual comparison.



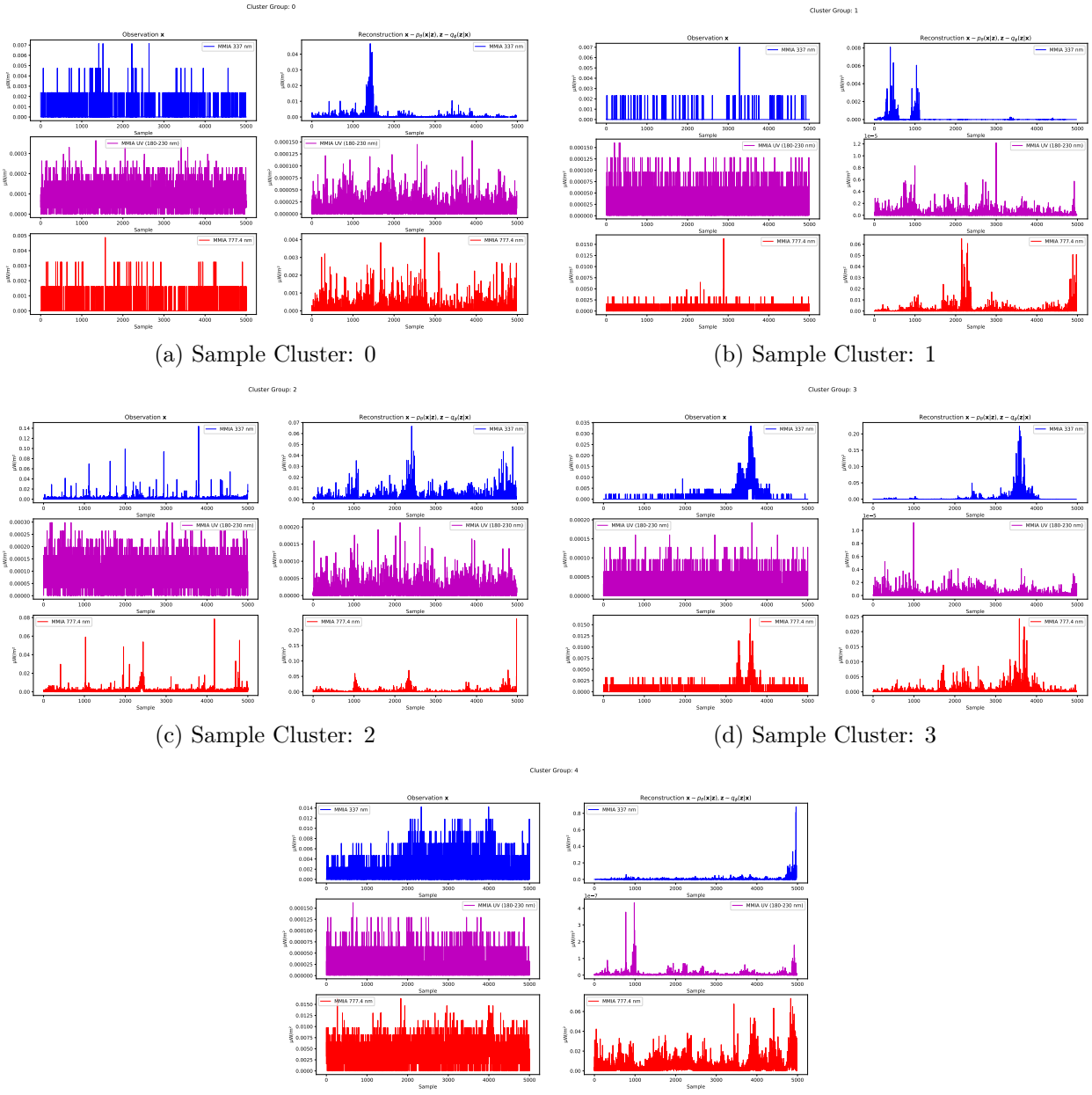
(a) Full reconstruction, including edges (boundaries). Left shows original data, $\mathbf{x}$, right shows reconstruction, $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}), \mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$



(b) Reconstruction with the edge not shown, giving a better visual comparison. Left shows original data, $\mathbf{x}$, right shows reconstruction, $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}), \mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$

Figure 6: A comparison of an event including the edges and excluding the edges. This shows the boundary issue, especially prominent in the UV spectra in this specific case, where spikes at the boundary creates problems with visual comparison of the original data and the reconstruction.

With this in mind, a sample from each cluster can be drawn, and plotted along side the original timeseries, as shown on fig. 7. These are ment as single samples to inspect how well the CVAE is able to reconstruct different categories from the dataset. For instance from the sample of Cluster 3, the CVAE appear to fare well on well-defined single peaks; capturing the overall structure and timing of the event. A more thorough discussion can be found in the Discussion section.



(a) Sample Cluster: 0

(b) Sample Cluster: 1

(c) Sample Cluster: 2

(d) Sample Cluster: 3

(e) Sample Cluster: 4

Figure 7: Samples from each of the clusters. This shows a side-by-side comparison of the original data, $\mathbf{x}$, along with its reconstruction, $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}), \mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$. The edges are not included due to the boundary issue, as described before.

5 Discussion

When testing the CVAE, the initial focus was on increasing the number of epochs, and implementing the WU, based on the results presented in [Sønderby *et al.*, 2016]. The first test, with the only change from the baseline being the WU effect during the first 100 epochs, showed an improvement in the certainty in the noise cluster of 20%, with the remaining clusters being close to the baseline. As also demonstrated by [Memarzadeh *et al.*, 2020], their CVAE is strong at detecing anomolous data, so intuitively it makes sense that the same is the case here. It was chosen to move forward with increasing the number of epochs, and WU effect to 2/5 of training, which again increased the noise cluster from 57% to 66%. With this in mind, it seems that both increasing epochs and WU so far increases the ability to detect noise. However, increasing the number of epochs to 800, from 500, but decreasing the WU time down to 1/4'th of training, dropped hte noise detecting significantly, down to 43%. This does appear to indicate, that a lower limit of WU time exists. It is suspected that this is because it takes a longer time to learn the importance of nodes in deeper layers, and if the WU effect is turned off too early, these still end up pruned away. Therefore, increasing the WU effect to 50% of training, and keeping the number of Epochs constant was chosen. This showed a significant increase in performance, boosting the noise clusters percentage up to 74%. It is speculated that perhaps higher fractions of WU could further improve the results, but there will have to be an upper limit. The CVAE should converge during its training, so there must be a natural upper limit to how long the WU periode can last in order to still achieve convergence during training.
Through all tests no real improvement in any other categories have been observed unfortunetly, only sporadic changes in the 30%-50% interval. Therefore, the performance of the noise cluster has been the main parameter when scoring a test. Due to time limitations, it was chosen to look into the effects of learning rate at this point. Doubling the learning rate showed an increase again to 87%, whereas decreasing dropped it significantly. However, doubling the learning rate also lead to more unstable training, requiring multiple initilizations before achieveing stable training. It makes sense that by lowering the learning rate, in order to achieve better results, it is necessary to also increase the number of epochs further, but this has not been tested, again due to time limitations. However, the unstable training was suspected to be due to the *dying relu* problem, especially since unstable training was more prominent with higher learning rates. Therefore, tests of implementing the LeakyReLU activation function, instead of ReLU, was used. Suprisingly, this yielded worse results, and training with a learning rate of 2e-4 was impossible, which is why no such inclusion is present in table 1 for the LeakyReLU activation function. A compromise was made, and training with a learning rate of 1.5e-4 was used, but the results where far from the earlier results with ReLU activation function. This could indicate that the problem is not with the dying relu issue, but instead perhaps an initilization problem. Unfortunetly, different types of initilizations where not tested in this project.
The mean distance to the cluster center (MDCC) for each specific cluster was also computed. This was based on an intuitive guess that a well resolved cluster would have a lower mean distance. This does not appear to be the case, when comparing different iterations of the CVAE; for instance the best performing noise cluster has the second highest MDCC of the 4 highlighted noise clusters in table 1. However, the MDCC of the noise clusters with respect to the remaining clusters in a given test appear to be among the lowest. This could indicate that MDCC is a bad parameter to use when comparing different tests of different parameters of the CVAE, but a good parameter when internally comparing the clusters to each other of a given test.
The way the score is computed for a given cluster does have some significant limitations, which are important to consider. For instance, since the label of a cluster is based on which of the 5 classes is overrepresented, but does not consider the overall total, there is no guarentee that all 5 clusters are actually represented. This is also the case when looking at table 1. It also means that if a cluster is very small, and contains very few observations, it is easy to score a very high percentage. For instance, looking at the 2D projection of the latent space on fig. 5, it is clear that clusters 0, 1 and 4 has significantly fewer observations than 2 and 3. Cluster 0 which was labelled as Noise with a certainty of 87% only contains 31 true occurences of noise, out of the total of 95 in the entire ground-truth dataset. This means that it hits only approximately 1/3 of all the noise in the ground-truth dataset, but hits this 1/3 with a high certainty. However, all groups are above 20%, which would

be the absolute possible minimum with 5 classes and 5 clusters, so it does better than blindly guessing. Clusters 1 and 0 also appear to be located very close to each other in the latent space, and both have the noise category, which is reassuring as that region of the latent space does indeed appear to belong to the Noise classification then.
One more thing to be aware of, is how the ground-truth dataset was made. This did not exist beforehand, and was created based on the baseline CVAE. It was a visual estimation of which 5 clusters could reliably be created. For the future, a more rigorous approach to creating the ground-truth dataset is needed. Some clearly defined classifications are needed; the classifications are based on visual estimates of shared features right now, and are suspect to individual bias when classifying the ground-truth dataset. It is noteworthy that the most well defined cluster is the Noise cluster, as this is also by far the easiest to visually determine, and therefore most likely the least suspect to personal bias. However, since both clusters 2 and 3 are so large and spread out as seen from fig. 5, there are also definitive problems with resolving these clusters, and the problem does not only appear to be with the ground-truth dataset itself.
The boundary issue also persists in the reconstructions, as seen from fig. 6. It is suspected to arrise from the way that the slicing of the dataset is done in preprocessing. Since the slicing does not take into account whether it is slicing in the middle of a peak or not, it is possible that a large amount of training datasets are sliced at a peak. This could skew some bias towards very rapid changes at the boarders of the slices. However, this is not confirmed in any way, and is just a suspicion for now.
Finally, the reconstruction itself is interesting look at, as seen from fig. 7. A single random sample from each cluster of the latent space is shown here, along with its original dataset. Clusters 0 and 1 were both classified as Noise, and both these samples are fine examples of noisy data; no overall structure, and very weak signal strength. The reconstruction clearly shows that the CVAE attemps to make sense of the noise, and predicts some features to be present. This is in agreement with the general idea behind anomaly detection with CVAEs, as also described by [Memarzadeh *et al.*, 2020]. The CVAE has seen less anomalous data during training, and therefore expects to find patterns, and tries to reconstruct them. In [Memarzadeh *et al.*, 2020], the reconstruction loss is used as the basis for detecting anomalous data, but the clustering used in this report does appear to achieve the same to some degree.
For cluster 2, this was predicted to be HCLA, with only a small majority of 27% of the cluster. This specific sample does not appear to agree with that classification, and would be classified as LCHA, as very little correlation seems to be present, and instead is dominated by sporadic high activity. This is also one of the two large groups in the latent space that are still not well defined, so it is to be expected that the classification and reconstruction is still not well resolved. The sample from Cluster 3 shows a very well defined peak though, which the CVAE appear to easily reconstruct. This is reassuring, as these types of events are most likely some of the most common classic lightning events, and most likely in the majority. Lastly, for the sample from cluster 4, the CVAE appear to find some structure in the signal. This seems to be a very weak and noisy signal, which makes it hard to validate whether this is correct or not. Interestingly, a lot of features are seen in the UV spectrum, which was a big problem for the baseline CVAE, discussed briefly in [Jørgensen & Engell, 2020]. Cluster 4 is also a small, well defined cluster, similar to Cluster 0 and 1. However, the label is poorly defined, with HCLA making up 34% of the cluster. This might be a case of problems in the ground-truth dataset, where this specific Cluster shares some features that were not considered when making the ground-truth dataset.
Overall, the CVAE appear to perform very similar to that proposed by [Memarzadeh *et al.*, 2020]. It seems to mainly distinguish between noise and nominal data, which is also a good start. With the goal being to lessen the burden of manually reviewing the MMIA data, being able to remove the noise is a good start, even though this is also the easiest group to manually determine.

6 Conclusion

In order to lessen the burden of manually reviewing data from the MMIA instrument, onboard the ASIM mission, the goal has been to optimize a CVAE, building ontop of the one presented in [Jørgensen & Engell, 2020]. In order to do this, a ground-truth dataset has been developed, which can be used in order to score different iterations of the CVAE, in order to determine the best performing CVAE. The score is based on the percentage of classes in a given cluster, and the mean distance to the cluster center (MDCC). The MDCC proved to be insufficient when comparing different iterations of the CVAE, but seemed to be a good indicator when comparing clusters internally in a test.
The optimization planed to specifically test the effect of longer training periods, the effects of adding warm-up, as presented by [Sønderby *et al.*, 2016], different activation funtions, and changes in the learning rate. The best performing CVAE was trained for 800 epochs, with a warm-up effect during 50% of training, using ReLU activation functions, and a learning rate of 2e-4. This iteration of the CVAE proved efficient in detecting noise specifically, with cluster 0 containing roughly 1/3 of the noise in the ground-truth dataset, with a 87% certainty. The remaining 4 categories are generally poorly resolved, with certainties in the 25%-50% range. This is still better than blindly guessing, but not reliable for finding specific categories. This would indicate that the CVAE appears to work very similar to the one presented by [Memarzadeh *et al.*, 2020], which also specifically looks to find anomalous data in their dataset. However, since the CVAE still only detects 1/3 of the noise, there are definitive room for improvement. The tests here would indicate that it is definitively possible to make a CVAE able to detect most noise in the dataset, it is only a question of finding the optimal version of the CVAE. The architecture of the CVAE would also be an obvious choice to look into in the future, to see if that can improve performance.
The original CVAE presented by [Jørgensen & Engell, 2020] discusses the problem of the boundary problem in the reconstructions of the CVAE. This problem remains unresolved, and is speculated to be due to the way the slicing of the data is done in the preprocessing, which might cause bias towards sharp changes in the edges. However, this is specualation at present time, and has not been determined with certainty.
Besides the boundary issue, the reconstructions are behaving as one would expect; the correlated, well defined peaks, corresponding to normal lightning, are easily encoded and decoded by the CVAE. These are expected to be among the most common events, with the most well defined features, so this is to be expected. The noise events are poorly reconstructed, which makes sense, assuming these are less common in the training data. An approach similar to [Memarzadeh *et al.*, 2020], where using the reconstruction loss as a criteria for noise, is most likely possible, but has not been tested; instead simply clustering the latent space is utilized. Events that are not clearly normal lightning or noise have a poorer reconstruction, and appear to behave as a middle ground between the very well resolved clear lightning events, and the poorly reconstructed noisy data. These are also hard to physically interpret at current time, so no clear conclusion on the reconstruction is made as of now. For that, it is necessary to better understand the physical nature, in order to figure out if the reconstruction makes sense, such an example is shown on Sample Cluster 2 on fig. 7.

References

Chanrion, Olivier, Neubert, Torsten, Lundgaard Rasmussen, Ib, Stoltze, Christian, Tcherniak, Denis, Jessen, Niels Christian, Polny, Josef, Brauer, Peter, Balling, Jan E., Savstrup Kristensen, Steen, Forchhammer, Søren, Hofmeyer, Peter, Davidsen, Peter, Mikkelsen, Ole, Bo Hansen, Dennis, Bhanderi, Dan D. V., Petersen, Carsten G., & Lorenzen, Mark. 2019. The Modular Multispectral Imaging Array (MMIA) of the ASIM Payload on the International Space Station. *Space Science Reviews*, **215**(4), 28.

ESA. 2018. *Electrical discharges in the atmosphere*. `http://www.esa.int/ESA_Multimedia/Images/2018/04/Electrical_discharges_in_the_atmosphere`.

Goodfellow, Ian, Bengio, Yoshua, & Courville, Aaron. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Ioffe, Sergey, & Szegedy, Christian. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, **abs/1502.03167**.

Jørgensen, Rasmus Christian, & Engell, Søren Christian Winther. 2020. *Transient Luminous Event Detection with Unsupervised Convolutional Variational Auto-Encoder*. `https://github.com/RCJoergensen/Synthesis_CVAE_ASIM/tree/main/Literature`.

Jørgensen, Rasmus Christian, & Olesen, Emilie Petrea Petajamaa Wiinberg. 2020. *Analysis of blue lightning measured by ASIM*. https://findit.dtu.dk/en/catalog/2491839329.

Kingma, Diederik P., & Welling, Max. 2019. An Introduction to Variational Autoencoders. *CoRR*, **abs/1906.02691**.

Leung, Kenneth. 2021. *The Dying ReLU Problem, Clearly Explained*. https://towardsdatascience.com/the-dying-relu-problem-clearly-explained-42d0c54e0d24.

Memarzadeh, Milad, Matthews, Bryan, & Avrekh, Ilya. 2020. Unsupervised Anomaly Detection in Flight Data Using Convolutional Variational Auto-Encoder. *Aerospace*, **7**(8).

Neubert, Torsten, & Chanrion, Olivier. 2020. Personal Communication.

Neubert, Torsten, Østgaard, Nikolai, Reglero, Victor, Chanrion, Olivier, Heumesser, Matthias, Dimitriadou, Krystallia, Christiansen, Freddy, Budtz-Jørgensen, Carl, Kuvvetli, Irfan, Rasmussen, Ib Lundgaard, Mezentsev, Andrey, Marisaldi, Martino, Ullaland, Kjetil, Genov, Georgi, Yang, Shiming, Kochkin, Pavlo, Navarro-Gonzalez, Javier, Connell, Paul H., & Eyles, Chris J. 2020. A terrestrial gamma-ray flash and ionospheric ultraviolet emissions powered by lightning. *Science*, **367**, 183–186.

Neubert, Torsten, Chanrion, Olivier, Heumesser, Matthias, Dimitriadou, Krystallia, Husbjerg, Lasse, Rasmussen, Ib Lundgaard, Østgaard, Nikolai, & Reglero, Victor. 2021. Observation of the onset of a blue jet into the stratosphere. *Nature*, **589**(7842), 371–375.

Nielsen, A. Michael. 2015. *Neural Networks and Deep Learning*. `http://neuralnetworksanddeeplearning.com/index.html`.

NIST. 2003. *Engineering Statistics Handbook - Beta Distribution*. `https://www.itl.nist.gov/div898/handbook/eda/section3/eda366h.htm`.

Sønderby, Casper Kaae, Raiko, Tapani, Maaløe, Lars, Sønderby, Søren Kaae, & Winther, Ole. 2016. How to Train Deep Variational Autoencoders and Probabilistic Ladder Networks. *In: Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*. JMLR: Workshop and Conference Proceedings. 33rd International Conference on Machine Learning (ICML 2016), ICML ; Conference date: 19-06-2016 Through 24-06-2016.

Szandała, Tomasz. 2020. *Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks*. Singapore: Springer Singapore. Pages 203–224.

Ygor, Rebouças Serpa. 2020. *A Comprehensive Guide on Activation Functions*. https://towardsdatascience.com/a-comprehensive-guide-on-activation-functions-b45ed37a4fa5.