Karlsruhe Institute of Technology

INSTITUTE OF INFORMATION SYSTEMS AND MARKETING

Biosignals Decoders: Improve available classifiers through data augmentation using generative AI

Seminar Thesis

Alessio Negrini

2106547

At the Department of Economics and Management
at the Institute of Information Systems and Marketing (IISM)
Information & Market Engineering

Supervisor: Dr. Michael Knierim

February 14, 2024

Abstract

In recent years, Machine Learning techniques have experienced significant growth, driven by advancements in computation power and data availability. Particularly, generative artificial intelligence (Gen AI) has become very popular due to Large Language Models, such as ChatGPT. In this seminar thesis, we implemented a Variational Autoencoder (VAE), which falls under the domain of Gen AI, and synthesized realistic electroencephalography (EEG) data. We then investigated the impact of synthetic data on the EEGNet classifier - a convolutional neural network tailored for EEG data. We found out that the low amount of realistic samples still poses a significant challenge, especially for data driven approaches such as Deep Learning. Collecting Electroencephalography (EEG) data remain a difficult, time consuming and error prone process. However, we were able to show that populating the training set with synthetic data created by our VAE, stabilized our training of the EEGNet and improved the generalization ability of the network. This finding holds promise for mitigating the difficulties associated with collecting EEG data.

Keywords— Variational Autoencoder (VAE), EEGNet, Brain Computer Interfaces (BCI), Electroencephalography (EEG), Classification, Deep Learning, Generative AI

Contents

List of Figures I

List of Tables II

1. Introduction 1

2. Related Work 2

3. Theoretical Introduction 3
- 3.1. Gen AI: Variational Autoencoder . . . 3
  - 3.1.1. Encoder-Decoder Architecture . . . 3
  - 3.1.2. Loss Function of an VAE . . . 4
- 3.2. EEGNet: A compact CNN for EEG-based BCIs . . . 5

4. Methodology 6
- 4.1. Preprocessing . . . 6
- 4.2. Implementation of our VAE . . . 6
- 4.3. Implementation of the EEGNet . . . 7
- 4.4. Experiment Design . . . 8

5. Results 9
- 5.1. Results of the VAE . . . 9
- 5.2. Results of the EEGNet . . . 9

6. Discussion 11

7. Conclusion & Outlook 12

8. Declaration 13

Appendix 14
- A. Architecture of the VAE . . . 14
- B. Latent Space Visualization . . . 15
- C. Reconstructions Example . . . 16

References 17

List of Figures

1.1. The Field Study Design that we used to collect EEG data from voluntary participants. In the following work, we will focus on the mental arithmetic round 'Math Hard'. The participant provided mental workload feedback using a TLX Survey that serve as our classes. *(Source: Working Paper by Dr. Michael Knierim (KIT))* . . . 1

3.1. Example of a Variational Autoencoder (VAE) with the multivariate Gaussian assumption. The encoder maps to a mean and standard deviation vector representing the parameters of a multivariate Gaussian distribution. Backpropagation is done using the reparameterization trick. The decoder takes the sample latent vector z and decodes it back to the original data space. (Source: Weng (2018), Accessed 31.01.2024) . . . 4

4.1. Measurement of the effect on the EEGNet: We fitted the EEGNet 25 times with various amount of synthetic data added to the augmented training set to compute the mean and standard deviation of each configuration. *(Source: Authors Illustration)* 8

5.1. Learning curves of the Variational Autoencoder: As we set $\beta = 0.01$ we emphasized the reconstruction loss rather than the Kullback-Leibler divergence. The VAE, however, still learns well with a validation loss slightly higher than the train loss. *(Source: Authors Illustration)* . . . 9

5.2. Aggregation of the (Val) loss and (Val) accuracy of the original train set: Training is unstable and yield a very high standard deviation. *(Source: Authors Illustration)* 10

5.3. Aggregation of the (Val) loss and (Val) accuracy of the train set augmented with 50% of reconstructed data: Training has become more stable with significantly less variation in the validation accuracy. *(Source: Authors Illustration)* . . . 10

B.1. 2D reduced latent space using PCA. The input data yield different embeddings and form clusters in the latent space. We can create completely new samples by interpolating within this space, or create similar data to the input data by sampling from the formed clusters and feed it to the decoder. *(Source: Authors Illustration)* . 15

C.1. Reconstruction of sample X_{21} with shape $(250, 7)$ yields a Mean Absolute Percentage Error (MAPE) of around 5%. We can see that we are not able to fully reconstruct the data, due to the low amount of training samples and the complex multivariate structure of the EEG data. *(Source: Authors Illustration)* . . . 16

List of Tables

4.1. Instantiated EEGNet architecture with significantly less parameters due to the low amount of samples that we were able to collect. 7

5.1. Aggregated data of the last epoch (#32) of the different configurations: Adding 50% reconstructed data to the training set yielded the most promising results as the validation accuracy is the highest with a significantly lower standard deviation. . . 10

A.1. Encoder-Decoder architecture used in our VAE: A simple sequencing of Dense layers with Batch Normalization. 14

1. Introduction

As collecting electroencephalography (EEG) data remain a difficult and time consuming process there is a significant need for creating realistic EEG data to leverage data hungry techniques such as Machine Learning. Multiple Deep Learning frameworks have emerged during the recent years, such as Generative Adversarial Networks (Goodfellow et al. (2014)) and Variational Autoencoders (VAEs) introduced by Kingma and Welling (2022). Those frameworks can be arbitrary tailored for the specified use case. Hartmann, Schirrmeister, and Ball (2018), for example, created the EEG-GAN tailored for BCI data. In this work, however, we implemented a Variational Autoencoder which pose a potential solution to mitigate the challenge of collecting EEG data by learning a latent distribution which is able to produce realistic samples. We can then use those realistic samples to improve available classifiers, such as the EEGNet.

In the following work, we focused on the EEG data of one particular participant that we recorded over 10 sessions while doing difficult arithmetic tasks. We did so because EEG data has a high variance due to its complex nature. Factors, such as sleep, the person being recording and the stress level, all influence the EEG data and thus making it more difficult to extract meaningful patterns. Therefore, we are operating on an individual basis to create personalized models. In Figure 1.1 we depicted the Field Study Design that we used to collect EEG data. We extracted the mental workload of the participants using the NASA Task Load Index (TLX Survey). The results from the survey were used for the classification task, that we will explain later in this work.

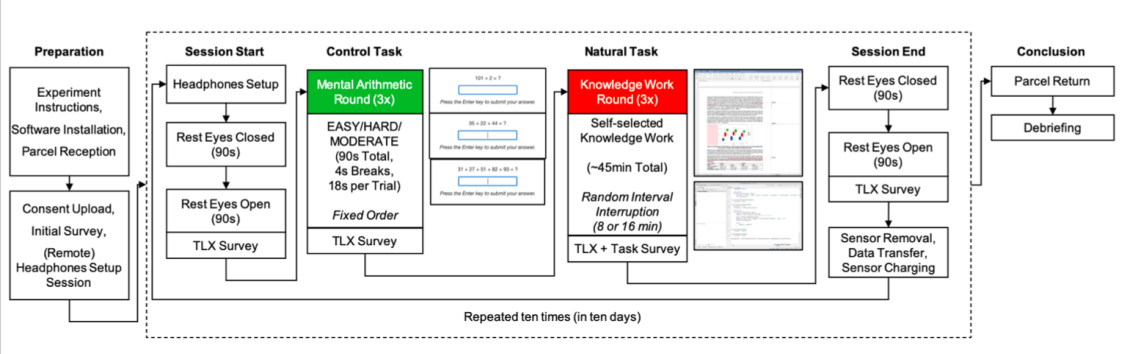

Figure 1.1.: The Field Study Design that we used to collect EEG data from voluntary participants. In the following work, we will focus on the mental arithmetic round 'Math Hard'. The participant provided mental workload feedback using a TLX Survey that serve as our classes. *(Source: Working Paper by Dr. Michael Knierim (KIT))*

In the following sections, we will first discuss the theoretical foundation of the generative AI method that we used. We will explain the core concepts behind Variational Autoencoders and the EEGNet that we used for classification. We will then dive deeper into the concrete methods that we implemented and explain our implementations of the VAE and the EEGNet. Last but not least, we present our results and give an outlook for future works.

2. Related Work

We used the framework of Vartional Autoencoder (VAE) for our generative model that was introduced by Kingma and Welling (2022). In their work, they introduced a stochastic variational inference and learning algorithm that scales to larger datasets. The resulting estimator can be optimized using standard stochastic gradient ascent techniques and is used for efficient approximate posterior inference, even though the continuous latent variables and/or parameters have intractable posterior distributions. Kingma et al. also introduced the so called reparameterization trick, which makes VAEs end-to-end trainable.

Higgins et al. (2017) added a modification to the Variational Autoencoder framework proposed by Kingma and Welling (2022): They introduced an adjustable hyperparameter β that balances latent channel capacity and independence constraints with reconstruction accuracy. A higher β encourages more efficient latent encoding and encourages the feature disentanglement. However, it may create a trade-off between reconstruction quality and the extent of disentanglement (Weng (2018)).

For the classification model we used the EEGNet proposed by Lawhern et al. (2018) that represents a convolutional neural network tailored for Brain Computer Interfaces (BCI) data. In their work they introduced depthwise and separable convolutions to create a model that is specific for EEG data and which encapsulates well-known EEG feature extraction. They also showed that their EEGNet generalizes across several BCI paradigms better, in the context of limited training data, than their reference algorithms (such as P30 visual-evoked potentials, ERN, MRCP and SMR).

In the context of creating synthetic multivariate time series data (similar to EEG data), the work proposed by Desai, Freeman, Wang, and Beaver (2021) introduced TimeVAE - a Variational Autoencoder tailored for time series data. Desai et al. proposed a novel architecture for synthetically generating time-series data using VAEs that also allow interpretability, ability to encode domain knowledge and reduced training times. In contrast to many approaches that were using Generative Adversarial Networks (GANs), such as TimeGAN proposed by Yoon, Jarrett, and Schaar (2019), Desai et al. were able to successfully adhere to the temporal correlations in time series data.

Hartmann et al. (2018) implemented the so called EEG-GAN: A generative adversarial network for EEG data. They introduced a modification of the improved Wasserstein GAN to stabilize the training and investigated a range of architectural choices which are critical for time series generation. By using different metrics, such as the Inception Score, Frechet Inception Distance and the sliced Wasserstein distance, they showed that the EEG-GAN framework generated naturalistic EEG examples and thus open a range of new generative application scenarios, such as data augmentation and EEG super-sampling.

Furthermore Bethge et al. (2022) also proposed an implementation of a conditional Variational Autoencoder (c-VAE) (first proposed by Sohn, Lee, and Yan (2015)) to learn generative-discriminative representations from EEG data. Their experimental results demonstrated that their model EEG2Vec is suitable for unsupervised EEG modelling and classification based on the latent representation achieving a robust performance and a good resembling of generated EEG data to real EEG data.

3. Theoretical Introduction

In this chapter, we will discuss the theoretical foundation of Variational Autoencoders, which are a popular instance of generative models. We will then shed light on the classifier that we used to test the effect of synthetic EEG data, namely the EEGNet – a convolutional neural network.

3.1. Gen AI: Variational Autoencoder

In the following section, we will discuss the theoretical foundation of a Variational Autoencoder. We begin by explaining the Encoder-Decoder Architecture and how learning is done using the reparameterization trick. We then investigate the loss function used within an VAE and dive deeper into each of its components.

3.1.1. Encoder-Decoder Architecture

A **Variational Autoencoder (VAE)** is a neural network consisting of two components: An encoder and a decoder network. The probabilistic **encoder** maps the input data to a mean and standard deviation vector that represent the parameters of a multivariate Gaussian distribution. This representation approximates the true (but intractable) posterior of the generative model $p_\theta(\boldsymbol{x}, \boldsymbol{z})$ (Kingma and Welling (2022):

$$log\ q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)}) = log\ \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\boldsymbol{I}) \tag{3.1}$$

The mean and the standard deviation of the approximate posterior, $\boldsymbol{\mu}^{(i)}$ and $\boldsymbol{\sigma}^{2(i)}$, are the corresponding outputs of the encoder network and are trained using the *reparameterization trick* (Kingma and Welling (2022)). The reparameterization trick allows backpropagating through the sampling process $\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$ by expressing the random variable $\boldsymbol{z}$ as a deterministic variable using an auxiliary independent random variable $\boldsymbol{\epsilon}$. A common choice for the approximate posterior is a multivariate Gaussian, as we can see in Equation 3.1. We can then express $\boldsymbol{z}$ via

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon},\ \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}) \tag{3.2}$$

with $\odot$ being the element-wise product.

The encoder network (usually a MLP) can contain, for example, Dense, Convolutional or LSTM layers. In this elaboration however, Dense layer were the most promising as it was less prone to overfitting. At this point, we kindly refer to the code base[1] to inspect the currently used architecture for the encoder and decoder.

The encoder of an VAE differs from vanilla Autoencoders (AEs) proposed by Hinton, Krizhevsky, and Wang (2011), since the encoder maps to the parameters of a Gaussian distribution. The AE, on the other hand, only passes the data through a fixed vector, the so called bottleneck layer and thus only compresses the input data. Vanilla Autoencoders do not allow sampling or interpolating, whereas in VAEs we can use the fitted approximate posterior to create synthetical data by doing so.

The second component of a Variational Autoencoder is the so called **decoder** network. It takes the sample from the latent distribution and reconstructs the original input data. As a measurement of difference between the original input $\boldsymbol{x}$ and the reconstruction $\hat{\boldsymbol{x}}$ we used the l_2-loss, i.e. the mean squared error (MSE).

[1] `https://github.com/negralessio/biosignals-gen-ai`

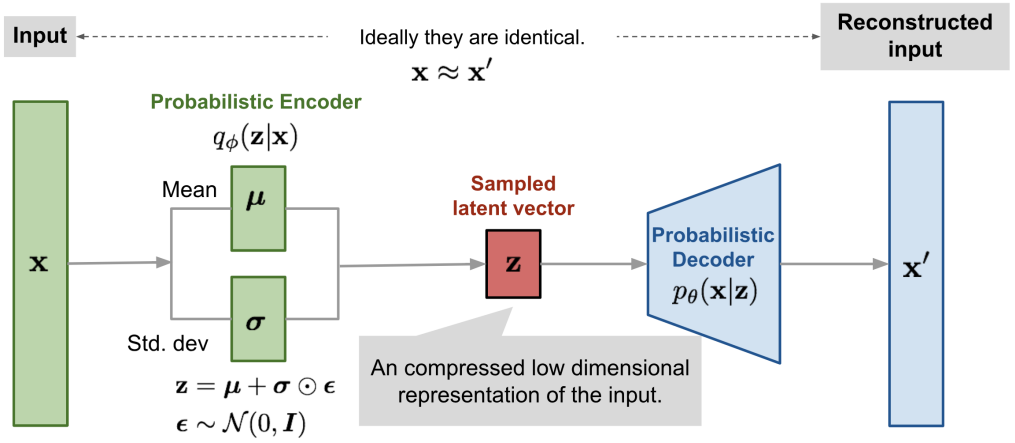

Figure 3.1.: Example of a Variational Autoencoder (VAE) with the multivariate Gaussian assumption. The encoder maps to a mean and standard deviation vector representing the parameters of a multivariate Gaussian distribution. Backpropagation is done using the reparameterization trick. The decoder takes the sample latent vector z and decodes it back to the original data space. (Source: Weng (2018), Accessed 31.01.2024)

3.1.2. Loss Function of an VAE

The loss function of our VAE $\mathcal{L}_{VAE}$ is composed of two different terms: A reconstruction loss $\mathcal{L}_{rec}$ and the Kullback-Leibler Divergence between the approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the prior $p_\theta(\boldsymbol{z})$, that we assume to be a standard normal distribution:

$$\mathcal{L}_{VAE} = \mathcal{L}_{rec} + \beta \cdot D_{KL}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z})\big) \tag{3.3}$$

The **reconstruction loss** forces the network to reconstruct its input as good as possible. In our implementation we used the l_2-loss, i.e. the mean squared error (MSE). The input is denoted as $\boldsymbol{x}$, whereas the reconstruction is denoted $\hat{\boldsymbol{x}}$:

$$\mathcal{L}_{rec} = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2^2\big] \tag{3.4}$$

The second term in $\mathcal{L}_{VAE}$ is the so called **Kullback-Leibler Divergence**, that was first proposed by Kullback and Leibler (1951). It measures how one probability distribution P differs from a second, so called reference probability distribution, Q. In the case of Variational Autoencoder, we want the approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to be as close as possible to the reference distribution $p_\theta(\boldsymbol{z})$, that we assume to be standard normal distributed. The KL-Divergence in the continuous case is defined as follows (Perez-Cruz (2008)):

$$D_{KL}(P \,||\, Q) = \int_x p(x)\ log\Big(\frac{p(x)}{q(x)}\Big)dx \tag{3.5}$$

Note that $D_{KL} = 0$ if the distributions are similar almost everywhere and that $D_{KL} \geq 0$ for any distributions. The KL-Divergence is also asymmetric, meaning that $D_{KL}(p||q) \neq D_{KL}(q||p)$.

Kingma and Welling (2022) have shown that, if both the prior $p_\theta(\boldsymbol{z}) = \mathcal{N}(0, \boldsymbol{I})$ and the posterior approximation $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ are Gaussian, the KL-Divergence (seen in 3.5) can be computed in a closed

form expression (see appendix B of Kingma and Welling (2022)):

$$D_{KL}(q_\phi(z|x) \,||\, p_\theta(z)) = -\frac{1}{2}\sum_{k=1}^{K} 1 + \log\sigma_k^2 - \sigma_k^2 - \mu_k^2 \tag{3.6}$$

Where K is the dimensionality of the sampled coding vector z by the encoder, μ_k and σ_k being the k^{th} component of the mean and standard derivation of z.

The effect of the KL-Divergence term on the VAE is two folded: First, it encourages the latent space to be smooth and continuous. Nearby points in the latent space corresponds to similar data points in the original input space and thus promotes meaningful interpolations and generative capabilities Doersch (2021). Second, it acts as a regularization term and therefore tackles the problem of overfitting Kingma and Welling (2022).

The last component to explain in the total loss of our VAE $\mathcal{L}_{VAE}$ is the parameter β. Higgins et al. (2017) augmented the original VAE framework with the single hyperparameter β. It controls the trade-off between the fidelity of data reconstruction and the disentanglement of learned representations in the latent space. If $\beta > 1$, the model is pushed to learn a more efficient latent representation of the input data and therefore puts more effort on the KL-Divergence with the risk of sacrificing reconstruction accuracy. Lower values of $\beta < 1$ prioritizes the reconstruction loss, but may sacrifices the ability of the VAE to construct a meaningful latent distribution. In our case of creating synthetic EEG data, we set $\beta = 0.01$ to emphasize accurate reconstructions of the input data.

3.2. EEGNet: A compact CNN for EEG-based BCIs

The EEGNet is a convolutional neural network classifier that surpasses methods for manual feature extraction by learning features in an end-to-end fashion. The network uses temporal convolution to learn frequency filters. It then uses a depth wise convolution connected to each feature map individually, to learn frequency-specific spatial filters. A temporal summary is learnt by separable convolutions followed by a pointwise convolution, which learns how to optimally mix the resulted feature maps together (Lawhern et al. (2018)).

The EEGNet, a compact CNN architecture for EEG-based BCI, was originally trained on 500 epochs with the categorical cross-entropy loss function (Lawhern et al. (2018)). It used the Adam optimizer, first proposed by Kingma and Ba (2017), which is currently the most used optimizer for backpropagation.

Lawhern et al. (2018) showed that their EEGNet generalizes better across several BCI paradigms and achieves comparably high performance to multiple reference algorithms when only limited training data is available. The used architecture is also very compact with only a small number of parameters, making it computationally efficient and suitable for real-time applications. The tensorflow-keras implementation of the EEGNet by the Army Research Laboratory (ARL) can be found on Github[2].

[2] `https://github.com/vlawhern/arl-eegmodels`

4. Methodology

In the this chapter, we will discuss the concrete methods that we used to generate synthetical EEG data. We first begin by enlisting the preprocessing steps, such as the transformations that we applied on the raw data. We then discuss and explain the development process of the Variational Autoencoders (VAEs) and its effect on the EEGNet.

4.1. Preprocessing

As we can see in Figure 1.1, we collected the EEG data from one participant over 10 sessions while solving mental arithmetic rounds (Math Hard in our case). The raw EEG data is then preprocessed using a series of transformations, such as:

- Mean-centered
- 25 & 50 Hz line noise (ZapLine, Static 2 Components)
- 2-15 Hz Bandpass FIR
- Referenced to L4+R4
- TLX-Reports (Mental workload surveyed) scaled to [0, 1]

To feed the input data to our Variational Autoencoder architecture, we also had to process the data into the desired shape and apply further transformations. The *Preprocesser* class located in the code base[1] is the responsible module and handles those transformations. Within this class, we scaled the 7 features / channels to [0, 1] using MinMax scaling provided by scikit-learn (Pedregosa et al. (2011)) and also applied windowing (of size 250) to our EEG data. The result is a three dimensional numpy array in shape $(440, 250, 7)$, i.e. 440 samples, each of size $(250, 7)$. Therefore, we are looking at 7 channels (features) each with a timestep of 250 ($\approx$ 2 seconds sampling).

4.2. Implementation of our VAE

We implemented the Variational Autoencoder using the *tensorflow-keras* framework. As the Encoder and Decoder we used a Multi-Layer-Perceptron (MLP) consisting of Dense layers and Batch Normalization. We used a latent dimension of 8, meaning that our latent distribution is a 8 dimensional multivariate Gaussian. You can view the exact architecture in Table A.1

We also tried many different architectures by implementing the BaseVAE and let diverse other architectures (such as DenseVAE, ConvVAE and LSTMVAE) inherit from that base class. Those child classes only had to implement the encoder and decoder architecture in which we used the corresponding layer types (Dense, Convolutional and LSTMs (Hochreiter and Schmidhuber (1997))). However, as we have a very low amount of data, DenseVAE outperformed the other architectures.

Regarding the fitting process: We split the data into a validation set of size 20% and trained on 128 epochs with a batch size of 4. We also used the Adam optimizer using default parameters proposed by Kingma and Ba (2017). For the loss function we used the loss as discussed in Section 3.1.2, namely a composition between the mean squared error (MSE) and the Kullback-Leibler divergence, with a β of 0.01.

[1] `https://github.com/negralessio/biosignals-gen-ai/blob/master/src/preprocessing.py`

4.3. Implementation of the EEGNet

In order to test the synthetic data created by our VAE, we had to reformulate our initial data into a supervised problem. We conducted a TLX survey to determine the mental workload of our participant during the field study. The mental workload was then scaled between $[0, 1]$ and discretized into the following classes:

$$\text{class}(x) = \begin{cases} 0 & \text{if workload}(x) < 0.1 \\ 1 & \text{if } 0.1 \leq \text{workload}(x) < 0.3 \\ 2 & \text{otherwise} \end{cases}$$

This led to the class distribution of having 352 samples being class 2 (heavy mental workload) and 88 samples being in class 1 (medium mental workload). The above mentioned discretization was done by analysing the distribution of the mental workload using a histogram.

The EEGNet classifier was instantiated with a dropout rate of 0.5, with a significant lower filter size F_1 of 2 and a depth parameter D of 1 to learn spatial filters for each feature map. The number of pointwise filters F_2 was also set to 2. Overall, we decided to decrease the architectures parameter to fit more on the low amount of samples that we had and thus mitigating overfitting. Between the layers, we used the Exponential Linear Unit (ELU) activation function. A concrete overview of the architecture that was used can be seen in Table 4.1.

We split our data into a train (67%) and validation set (33%). We then compiled and fitted the model for 32 epochs with a batch size of 4 using the binary crossentropy loss and the Adam optimizer with default parameters. As an additional metric we used the accuracy.

Layer (type)	**Output Shape**	**Param #**
input_27 (InputLayer)	(None, 7, 250, 1)	0
conv2d_26 (Conv2D)	(None, 7, 250, 2)	32
batch_normalization_78	(None, 7, 250, 2)	8
depthwise_conv2d_26	(None, 1, 250, 2)	14
batch_normalization_79	(None, 1, 250, 2)	8
activation_52	(None, 1, 250, 2)	0
average_pooling2d_52	(None, 1, 62, 2)	0
dropout_52	(None, 1, 62, 2)	0
separable_conv2d_26	(None, 1, 62, 2)	36
batch_normalization_80	(None, 1, 62, 2)	8
activation_53	(None, 1, 62, 2)	0
average_pooling2d_53	(None, 1, 7, 2)	0
dropout_53	(None, 1, 7, 2)	0
flatten	(None, 14)	0
dense	(None, 2)	30
softmax	(None, 2)	0
Total params	136 (544.00 Byte)	
Trainable params	124 (496.00 Byte)	
Non-trainable params	12 (48.00 Byte)	

Table 4.1.: Instantiated EEGNet architecture with significantly less parameters due to the low amount of samples that we were able to collect.

4.4. Experiment Design

To evaluate the effect on the EEGNet using synthethic data from our VAE we conducted the following experiment: First we split the data into a train and validation set with a 3 : 1 ratio. This was done using a fixed random state, to ensure that in each run we have the same validation set. We then called the fitting method of our EEGNet in total 25 times, while we first added 0% reconstructed data. Then we repeated the process, but this time we added 25% reconstructed data to our augmented train set. The reconstructed data was used by calling the .predict() function of our, on all available data fitted, Variational Autoencoder. This process was then repeated again for 50% and 100% added reconstructed data. In each run we collected the history data of our fitting, namely the losses (for train and validation set) and the accuracy. The experiment design can be seen in Figure 4.1.

We used the reconstructions instead of sampling from the prior and feeding it to the decoder for simplicity reasons. Additionally, we emphasized the reconstruction error more than the Kullback-Leibler divergence by setting β to 0.01 meaning that the KL loss remain higher. Therefore, the samples created by sampling from the prior hold higher risk not being realistic or meaningful. However, as we were not able to fully reconstruct the input data (as seen in the following chapter), using the reconstructions has almost the same effect as creating synthetic data by sampling from the prior. Due to using the reconstructions rather than sampling from the prior, we also know the corresponding classes, i.e. the class from the original sample that we are trying to reconstruct.

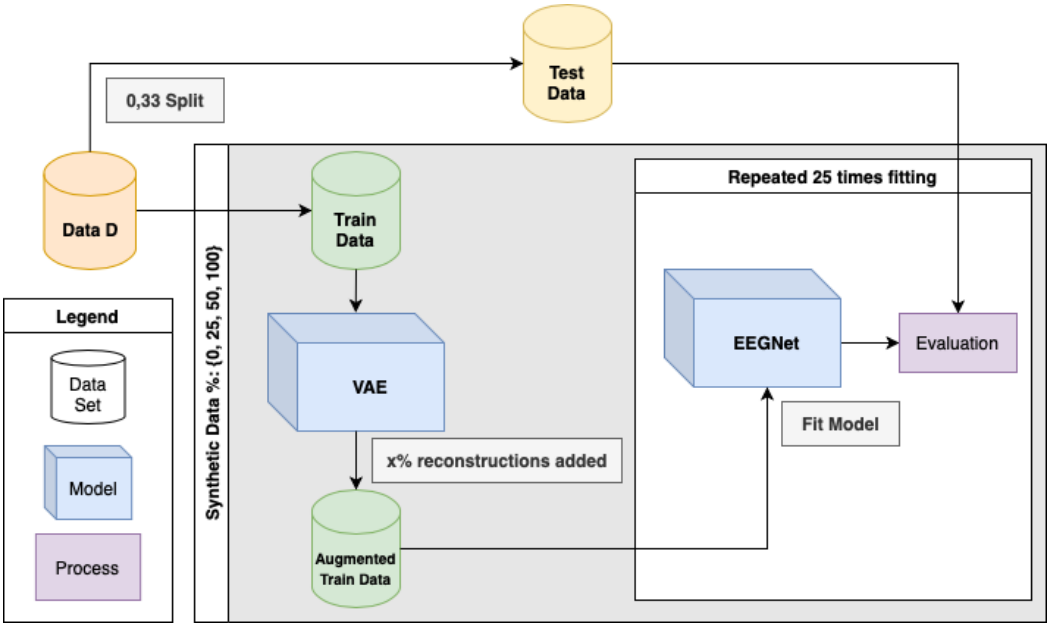

Figure 4.1.: Measurement of the effect on the EEGNet: We fitted the EEGNet 25 times with various amount of synthetic data added to the augmented training set to compute the mean and standard deviation of each configuration. *(Source: Authors Illustration)*

5. Results

In this chapter, we will discuss the results of our Variational Autoencoder and the effect of the resulting augmented data on the EEGNet. We begin by looking at the learning curves of the VAE and investigate the reconstructions as well as new synthetic data sampled from the latent space. Finally, we will discuss the effect on the EEGNet classifier and compare the different level of synthetic data added.

5.1. Results of the VAE

The learning curves of the VAE (Table A.1) can be seen in Figure 5.1. As we set β to 0.01 and thus emphasizing the minimization of the reconstruction loss (MSE), we successfully reduced it. Overall the VAE learns well and the validation loss is just slightly larger than the training loss as expected. However, due to the low amount of samples, we cannot reduce it any more and also the Kullback-Leibler divergence could be lower.

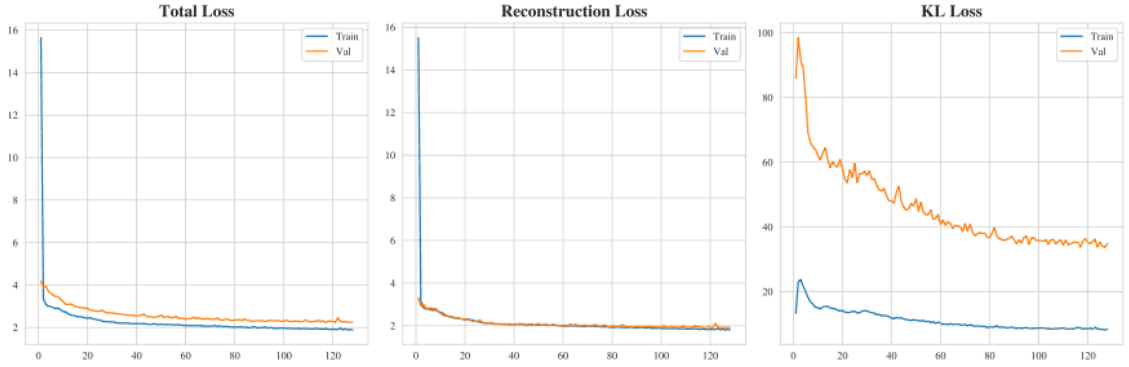

Figure 5.1.: Learning curves of the Variational Autoencoder: As we set $\beta = 0.01$ we emphasized the reconstruction loss rather than the Kullback-Leibler divergence. The VAE, however, still learns well with a validation loss slightly higher than the train loss. *(Source: Authors Illustration)*

After training, we analyzed the reconstructions, i.e. we provided our input train data to the VAE and measured the difference using the *Mean Absolute Percentage Error (MAPE)* and the MSE. An example of a reconstruction of sample X_{21} can be seen in Figure C.1, where we yield a MAPE of $\approx 5\%$. The average MAPE of all available data is around 15% and shows that we had difficulties reconstructing the input data. We assume that this is due to the low amount of samples and the rather complex multivariate structure of the EEG data.

We also reduced the dimensionality of the originally 8D latent space to 2 using Principal Component Analysis (PCA) and visualized it (Figure B.1). The input data form clusters in the latent space defined by the mental workload and the different mental workload classes are differently embedded by the encoder. There are patterns in the EEG data that yield similar embeddings. By interpolating between the clusters we can create completely new data by feeding it to the decoder.

5.2. Results of the EEGNet

We fitted the EEGNet classifier 25 times with various amount of synthetic data added to the train set (as seen in Figure 4.1) and computed the mean and standard deviation of the (val) loss and (val) accuracy.

The aggregation of the 25 fitting with 0% added reconstructed data can be seen in 5.2. The validation accuracy is very unstable with a significant high standard deviation (orange in the figure). Even in later epochs the validation accuracy may drop significantly.

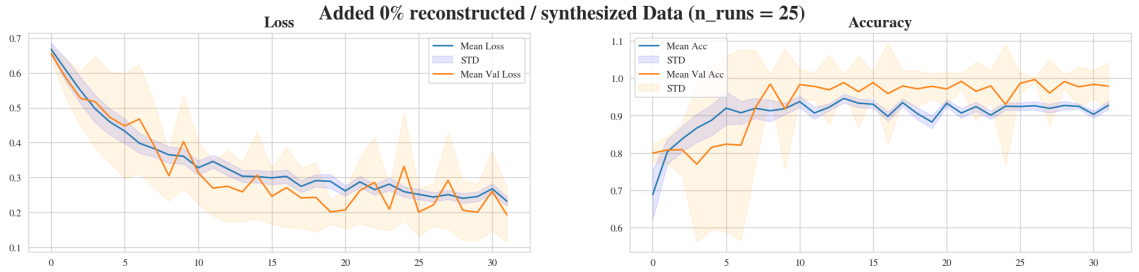

Figure 5.2.: Aggregation of the (Val) loss and (Val) accuracy of the original train set: Training is unstable and yield a very high standard deviation. *(Source: Authors Illustration)*

We also evaluated the EEGNet classifier with the augmented training set of adding 25, 50 and 100% of reconstructed data. The most promising result can be seen in Figure 5.3, where we achieved a significantly more stable training with less standard deviation in the validation accuracy.

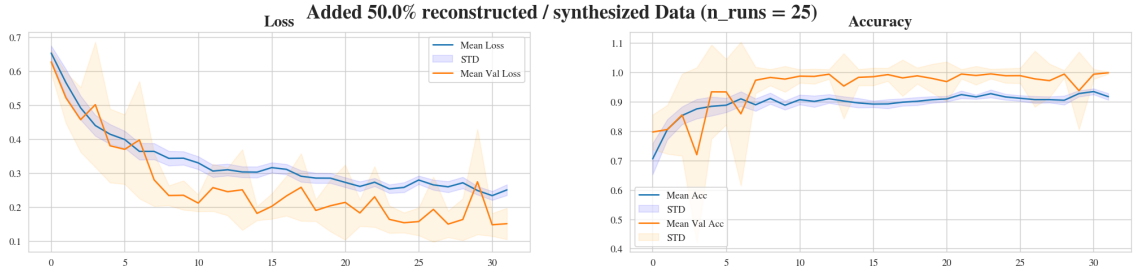

Figure 5.3.: Aggregation of the (Val) loss and (Val) accuracy of the train set augmented with 50% of reconstructed data: Training has become more stable with significantly less variation in the validation accuracy. *(Source: Authors Illustration)*

In table 5.1, we aggregated the loss and the accuracy of the last epoch (#32) over all runs. The actual sweet spot can be achieved by adding 50% of reconstructed data to the training set and use its augmentation to improve and stabilize the EEGNet classifier. Not only does it lowers the validation loss, but it also lead to significantly less deviation in the validation accuracy and thus improves the generalization ability of the EEGNet.

% Synth. Added	Mean Loss	SD Loss	Mean Val Loss	SD Val Loss
0%	0.2318	0.0129	0.1926	0.0777
25%	0.2934	0.0094	0.1789	0.0479
50%	**0.2512**	**0.0158**	**0.1518**	**0.0461**
100%	0.2101	0.0151	0.1420	0.0475

% Synth. Added	Mean ACC	SD ACC	Mean Val ACC	SD Val ACC
0%	0.9273	0.0113	0.9784	0.0625
25%	0.9130	0.0087	0.9942	0.0162
50%	**0.9173**	**0.0111**	**0.9986**	**0.0039**
100%	0.9524	0.0150	0.9956	0.0164

Table 5.1.: Aggregated data of the last epoch (#32) of the different configurations: Adding 50% reconstructed data to the training set yielded the most promising results as the validation accuracy is the highest with a significantly lower standard deviation.

6. Discussion

Due to the low amount of samples that we have for training the VAE (right now 440 samples per participant), our VAE is not fully able to reconstruct the input data. The shortage of data and the rather complex structure of the EEG data still pose a major challenge to overcome. Trying to synthesize multi channel time series data with several features and dozens of time steps remain difficult. Furthermore, it is demanding to evaluate synthetic time series data, whether it is a realistic sample or not. Unlike creating synthetic images, we cannot simply evaluate it by visualizing it. However, as seen above, adding reconstructed data (or even synthesized ones) does have a positive effect on Machine Learning approaches. Therefore, we were able to show that it is actually feasible to create personalized generative models and thus holds potential for future works.

The architecture of the VAE is also rather simple, as it contains only Dense layers followed by Batch Normalization (Table A.1). We tried different layers, such as LSTMs (Hochreiter and Schmidhuber (1997)), which are quite popular in handling sequence data and also convolutional layers. Nevertheless, in both cases LSTMs and 1D-CONV layers did not perform as good as initially expected. We assume that the low amount of data is the main driver. We proceed analogously with the EEGNet and lowered the amount of kernels inside the convolutional layers and thus reducing the amount of learnable parameters and mitigating overfitting. It is of future work to tweak the hyperparameters and search for the optimal architecture of the generative model, especially if we want to operate on a individual basis and create personalized models.

7. Conclusion & Outlook

Creating synthethic EEG data using Gen AI techniques, such as Variational Autoencoders, can be very beneficial for future studies regarding Brain-Computer-Interface (BCI) research. Training a generative model on a personal basis can be done to create realistic synthetical data to improve Machine Learning classifiers. However, generative models - or Deep Learning models in general - require a huge amount of data that need to be collected beforehand. This still remains a major issue, as collecting EEG data is a time consuming and difficult process.

In our study, we've demonstrated the feasibility of training a generative model on an individual basis, even with a limited amount of data. This approach has shown promising potential in enhancing other Machine Learning models, such as the EEGNet. In future work, it is important that more data is provided to improve generative models even better and thus allowing to create even more realistic samples. By having more data, we can build architectures that are able to learn a more complex distribution resembling the original distribution that creates our EEG data. Finding the perfect architecture and hyperparameters for creating a personalized generative model, such as the VAE, also remains time consuming and requires a lot of computation power.

8. Declaration

I truthfully declare that I have written this document independently, that I have completely and accurately indicated all aids used, and that I have marked everything that has been taken from the work of others, either unchanged or with modifications, and that I have complied with the KIT Statutes for Safeguarding Good Scientific Practice in the currently valid version.

Karlsruhe, February 14, 2024 _____

Alessio Negrini

Appendix

A. Architecture of the VAE

Layer (type)	Output Shape	Param #	Connected to
Encoder			
input_3 (InputLayer)	(None, 250, 7)	0	[]
flatten_1 (Flatten)	(None, 1750)	0	input_3[0][0]
dense_6 (Dense)	(None, 32)	56032	flatten_1[0][0]
batch_normalization_5	(None, 32)	128	dense_6[0][0]
dense_7	(None, 16)	528	batch_normalization_5[0][0]
batch_normalization_6	(None, 16)	64	dense_7[0][0]
dense_8	(None, 16)	272	batch_normalization_6[0][0]
z_mean (Dense)	(None, 8)	136	dense_8[0][0]
z_log_var (Dense)	(None, 8)	136	dense_8[0][0]
sampling_1 (Sampling)	(None, 8)	0	z_mean[0][0], z_log_var[0][0]
Decoder			
input_4 (InputLayer)	(None, 8)	0	[]
dense_9	(None, 16)	144	input_4[0][0]
batch_normalization_7	(None, 16)	64	dense_9[0][0]
dense_10	(None, 16)	272	batch_normalization_7[0][0]
batch_normalization_8	(None, 16)	64	dense_10[0][0]
dense_11	(None, 32)	544	batch_normalization_8[0][0]
batch_normalization_9	(None, 32)	128	dense_11[0][0]
decoder_final_dense	(None, 1750)	57750	batch_normalization_9[0][0]
reshape_1 (Reshape)	(None, 250, 7)	0	decoder_final_dense[0][0]
Total params		**116262**	
Trainable params		**116038**	
Non-trainable params		**126**	

Table A.1.: Encoder-Decoder architecture used in our VAE: A simple sequencing of Dense layers with Batch Normalization.

B. Latent Space Visualization

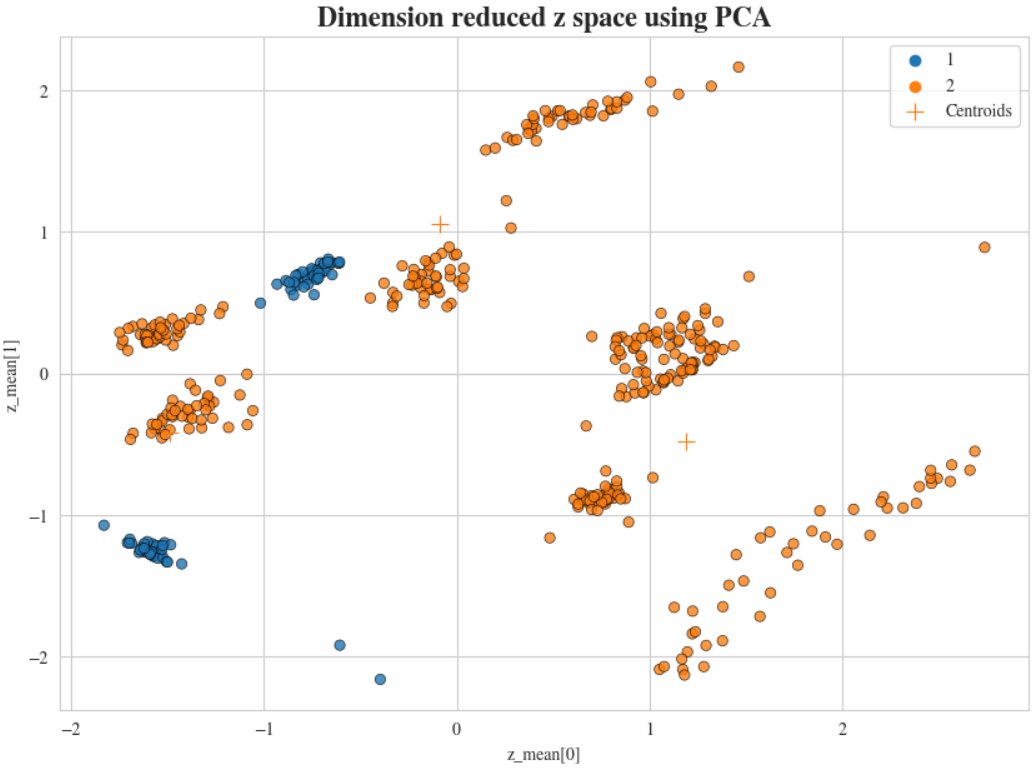

Figure B.1.: 2D reduced latent space using PCA. The input data yield different embeddings and form clusters in the latent space. We can create completely new samples by interpolating within this space, or create similar data to the input data by sampling from the formed clusters and feed it to the decoder. *(Source: Authors Illustration)*

C. Reconstructions Example

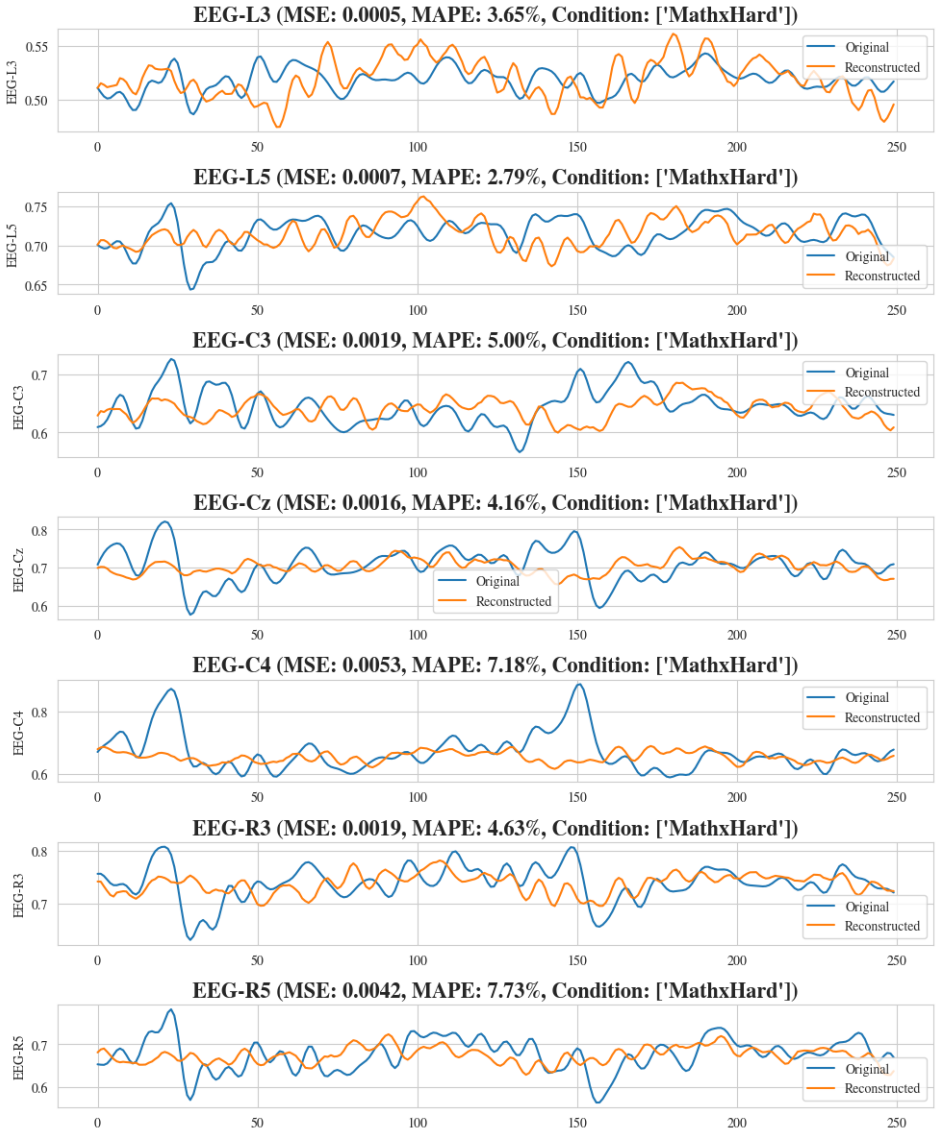

Figure C.1.: Reconstruction of sample X_{21} with shape $(250, 7)$ yields a Mean Absolute Percentage Error (MAPE) of around 5%. We can see that we are not able to fully reconstruct the data, due to the low amount of training samples and the complex multivariate structure of the EEG data. *(Source: Authors Illustration)*

References

Bethge, D., Hallgarten, P., Grosse-Puppendahl, T., Kari, M., Chuang, L. L., Özdenizci, O., & Schmidt, A. (2022). *Eeg2vec: Learning affective eeg representations via variational autoencoders.*

Desai, A., Freeman, C., Wang, Z., & Beaver, I. (2021). *Timevae: A variational auto-encoder for multivariate time series generation.*

Doersch, C. (2021). *Tutorial on variational autoencoders.*

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). *Generative adversarial networks.*

Hartmann, K. G., Schirrmeister, R. T., & Ball, T. (2018). *Eeg-gan: Generative adversarial networks for electroencephalograhic (eeg) brain signals.*

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., ... Lerchner, A. (2017). beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations.* Retrieved from `https://openreview.net/forum?id=Sy2fzU9gl`

Hinton, G. E., Krizhevsky, A., & Wang, S. D. (2011). Transforming auto-encoders. In T. Honkela, W. Duch, M. Girolami, & S. Kaski (Eds.), *Artificial neural networks and machine learning – icann 2011* (pp. 44–51). Berlin, Heidelberg: Springer Berlin Heidelberg.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Kingma, D. P., & Ba, J. (2017). *Adam: A method for stochastic optimization.*

Kingma, D. P., & Welling, M. (2022). *Auto-encoding variational bayes.*

Kullback, S., & Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79 – 86. Retrieved from `https://doi.org/10.1214/aoms/1177729694` doi: 10.1214/aoms/1177729694

Lawhern, V. J., Solon, A. J., Waytowich, N. R., Gordon, S. M., Hung, C. P., & Lance, B. J. (2018, July). Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. *Journal of Neural Engineering*, *15*(5), 056013. Retrieved from `http://dx.doi.org/10.1088/1741-2552/aace8c` doi: 10.1088/1741-2552/aace8c

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Perez-Cruz, F. (2008). Kullback-leibler divergence estimation of continuous distributions.

In *2008 ieee international symposium on information theory* (p. 1666-1670). doi: 10.1109/ISIT.2008.4595271

Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 28). Curran Associates, Inc. Retrieved from `https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf`

Weng, L. (2018). From autoencoder to beta-vae. *lilianweng.github.io*. Retrieved from `https://lilianweng.github.io/posts/2018-08-12-vae/` (Accessed on 2024-01-31)

Yoon, J., Jarrett, D., & Schaar, M. (2019, 12). Time-series generative adversarial networks..