Approaches to Synthetic EEG Data Generation

CS 4641 Final Project Report
Summer 2020

Rishi Banerjee
rbanerjee41@gatech.edu

Ishan Chadha
ichadha3@gatech.edu

Jakub Jackowiak
jjackowiak3@gatech.edu

I. Introduction

Electroencephalography (EEG) is a process that captures the electrical activity in the human brain. These electrical impulses show up as wavy lines plotted against time. Irregularities such as high peaks and valleys may be a sign of seizures or other brain disorders. A brain-computer interface (BCI) is a system that translates EEG recordings to external devices. Currently, brain computer interfaces pose a challenge to the technology community on numerous fronts, including but not limited to our lack of ability to derive useful interpretations from monitored brain activity [1]. At most, we only have partial and noisy multivariate data, which makes it difficult to construct meaningful information from current data.

If we are able to successfully access and interpret signals from the brain, projects such as Elon Musk's Neuralink will have an increased chance of developing fully functional brain computer interfaces [2]. Alongside this, by properly understanding the features of regular brain waves, we can pinpoint brain waves that are irregular and predict the early onset of neurological disorders [3]. The pipeline of a generic BCI system is shown below.

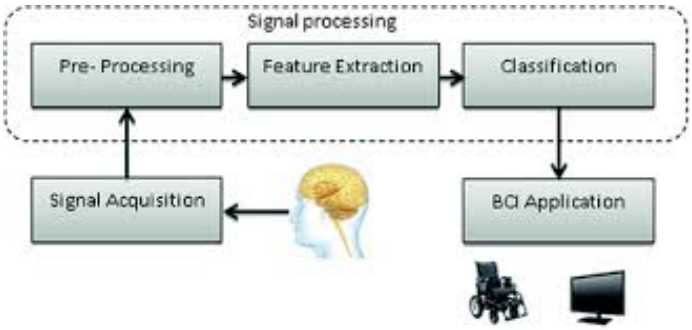

Fig. 1. Generic Brain Computer Interface System

However, there are many shortcomings in the process of collecting EEG signals in real-world environments, such noise impact the ability to derive good insights from the data and variance between subjects. As a result, many downstream applications of this technology would suffer from the unreliability of the data collected. As a result, the generation of realistic, generalizable training data could possibly circumvent this issues and increase effectiveness of downstream tasks [5].

In machine learning, generative models have long been used to generate entirely new and realistic data points which match the distribution of a given target dataset. Recent work on neural-based models such as Generative Adversarial Networks (GAN) have demonstrated that these are highly capable at capturing key elements from a diverse range of datasets to generate realistic samples. Increasingly, there is evidence that using synthetic data, taken from a generative model, can be used as a form of data augmentation to help improve the performance of any downstream data classification task

In our project, we explore the use of a variety of machine learning and deep learning algorithms to create realistic and generalizable EEG data. Our contribution is a novel angle for modeling of brain wave data using generative adversarial networks based on previously employed models such as hidden Markov models. We aim to have a model that is comparable to the performance of current brain wave generation machine learning architectures, allowing us to apply our work more broadly for applications mentioned in the background. This design also differs from prior studies in that we plan on generalizing the brain waves that we find across different subjects performing various activities with respect to the region of the brain being measured.

II. Background and Related Work

A. Generative Adversarial Networks

Generative models have been proven very powerful within the context of unsupervised learning where the model learns a hidden structure of the data from its distribution to generate new data samples within the same distribution. The creation of generative adversarial networks is shown in the seminal 2014 paper by Goodfellow [5]. Since then, GAN's have experienced wide success in rendering novel realistic images and image style transfer. The core of the framework is composed of two models, a generator and a discriminator. The generator (G) is trained to reproduce genuine target images from a specified input, while the discriminator (D) is trained to differentiate from generated images to naturally sampled images.

Generative Adversarial Networks have been previously used for reconstruction of high-sampling-sensitivity EEG signals from low-sampling-sensitivity EEG. Aznan et al found that the task of cross-subject generalization between subjects can increase by over 35 percent through the use of synthetic data created with neural-based generative models [4]. The Wasserstein distance can be used to improve the training of the model [6]. The EEG data varies heavily from person to person. Better results were achieved while using data from

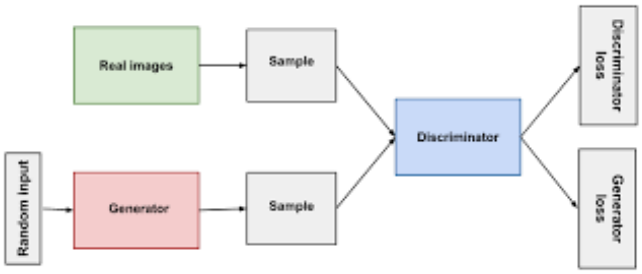

Fig. 2. Basic Structure of a GAN

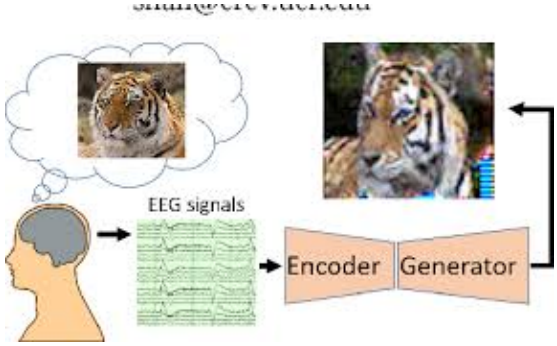

Fig. 3. Overview of Proposed Method of ThoughtViz

multiple subjects. Data from one hundred subjects gave the biggest improvements in the model performance with small improvements with more subjects [7].

B. Machine Learning with Brain-Computer Interfaces

Within the existing body of literature, there are instances of machine learning models to create synthetic EEG data and using EEG data to extract insights from the environment of the BCI.

For instance, [8] proposes using the improved WGAN with GP for a single channel of the EEG data for motor imagery task.

Instead of generating EEG signals, [9] generates previously seen images while having brain signals recorded by an EEG. Six subjects are shown 2000 images with 40 classes per subject. Using an LSTM model, the feature representations are encoded, which are subsequently used as condition vectors employed along with random noise vectors to generate new images. They obtain a test accuracy of 83.9% when evaluating the LSTM model for feature representation.

In the space of classifying and generating images and words from EEG data, one of the most commonly used datasets is the EEG-MNIST dataset from David Vivancos, which contains multichannel data with a classification of what number he was thinking of while being recorded [10]. This data is used to use the channel data as a classification problem. In addition to this, Tom Di Fulvio used the multiclass decision forest and the multiclass neural network algorithms to categorize certain brainwave patterns into words [11]. Similarly, Alberto Bozal classified images from signals generated by the viewer's brain using the dropout and the long short-term memory deep learning architectures [12]. Along with this, Praveen Tirupattur et al introduced the idea of ThoughtViz, a method of analyzing brain activity, recorded by an EEG, of a subject while thinking about a digit, character or an object and synthesize visually the thought item [13]. Below is a simplified version of the method shown in Tirupattur et al, which they use a Variational AutoEncoder model.

III. Design and Methods

A. Approach

We aim to have a model that is comparable to the performance of current brain wave generation machine learning architectures, allowing us to apply our work more broadly for applications. We believe that generalizability of data will allow us to have trained models without any required training time from the individual. Generalizability is important because forecasting generalizable data would be important for anomaly detection to see how far actual data deviates from the forecast. Along with this, for brain computer interface applications, having trained models based on generalizable data would lower the time that the device would need to be calibrated for downstream tasks. In order to do this, we will be choosing a subset of subjects and splitting them into training, validation, and testing groups. The success of the models is now dependent on the ability to find generalizable patterns since it is being fed no data from the subjects in the testing sets. We will test a variety of models to predict 3 future timesteps from up to 3 input timesteps. Normalized Root Mean Squared Deviation and Mean Average Percent Errors as evaluation metrics for 1-Day or multiple-day forecasts. A forecast would be thought of as successful if for each datapoint away from the input, the forecast is able to maintain the same values as the actual data.

B. Dataset and Feature Engineering

We used a subset of the TUH EEG Corpus dataset, collected by Temple University Hospital. It contains data from more than 13 000 patients. The EEG recordings have a sample frequency from 250Hz to 1024Hz with channels for 10/20 configuration. The data was stored in EDF+ file format, the typical data type for EEG data.

The first step we had to take to make the data usable was to convert the EDF files to CSV files. After that, we found break points in the data where the reading was zero for multiple time periods and we removed them from the dataset. There is no real-world advantage to the models detecting that there are break points for the problems that we are trying to solve for. After this, we needed to remove extraneous columns that detected the existence of anomalies. Since we were only forcasting, we did not need these columns and only used the 24 columns of real-valued data from the unique electrode channels. Next, to make this time series data usable for the different models, we created a supervised learning problem out of the data that we had. To do this, for each channel, we shift the series data mutiple time steps such that

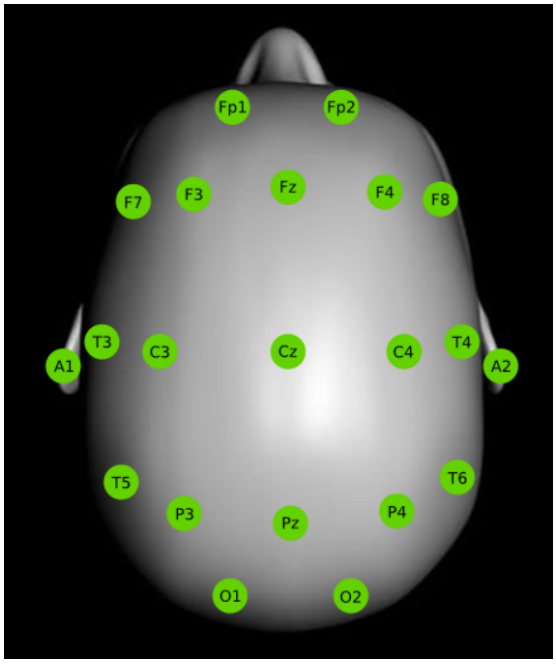

Fig. 4. Locations of Electrodes in 10/20 Configuration

for the data at time t, in line with it there will also be the data for $t-3, t-2, t-1, t+1, t+2$ for each channel. This was then reshaped so that the set of input data is of dimensions $(num\_samples, input\_steps, num\_channels)$ and the set of prediction data is of dimensions $(num\_samples, output\_steps, num\_channels)$.

C. Baseline: Support Vector Regression

First, we used SVR as a baseline model to find a line of best fit for whole recordings, which ranged up to hundreds of thousands of time steps.

Support vector machines (SVMs) were first developed around 30 years ago to tackle both linear and nonlinear classification problems using different kernel functions. The choice of this kernel function determines the type of nonlinearity that we attempt to use for correctly modeling our data. Support vector regression is a method developed using the same kernel trick for modeling nonlinearity but instead of classification where null space is maximized between clusters, null spaces is minimized between the regression line and the data points being modeled. This model was chosen because SVR's are commonly used to solve forecasting problems because of the ability to fit to very high amounts of data.

Due to the large amount of data for which each SVR was trained to model, the training itself took multiple hours, giving the following results for predicting the next 3 points.

D. SVR and HMM

In order to have more results in less time, we then decided to focus on creating smaller regression models in local regions of the wave. We ended up using samples of 100 points at a time to predict the next 3 points. However, we found that within

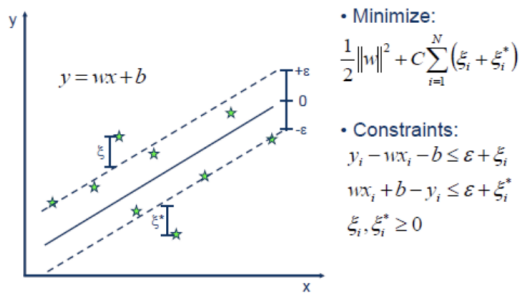

Fig. 5. SVR Optimization Problem

these 100 points, there was still a lot of variation in the type of nonlinear function that best modeled the data.

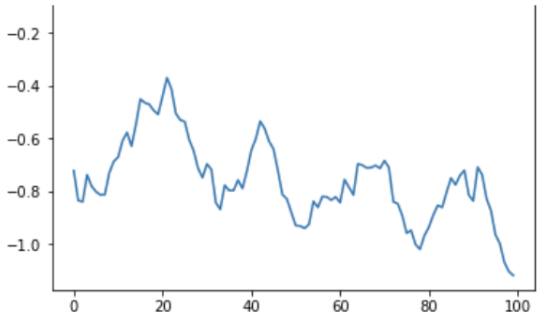

Fig. 6. EEG Sample of 100 Points

In order to further reduce the amount of points that the SVR was to regress on, we first sampled 100 points and then classified those points into distinct states. The border between each state roughly corresponded to breakpoints between differing nonlinear functions. After this, we only took the points from the final state of the classification and regressed on those using an SVR, followed by prediction of the next 3 points. For example, in Figure 6, we can clearly see that there are 4 peaks in the sample of 100 points shown. If we divide these 100 points into 4 distinct states and only regress on the final state, we will obtain a better model for predicting the next 3 points as opposed to regressing on all 100 time steps.

At first, we considered using Naive Bayes in order to classify samples of 100 points into distinct states, but the independence assumption of Naive Bayed led to very inaccurate splits of the data. We then decided to use Hidden Markov models, which are both lightweight and vastly more accurate.

Hidden Markov models assume that there is an underlying Markov chain which can be determined by observing the data, which in this case was EEG time step data. We know that there are some number of hidden states; for our data, 5 hidden states seemed to work best. We initialized the underlying Markov chain with arbitrary initial probabilities and transition probabilities, and then determined the most likely path that these Markov chain followed given the observations using the Viterbi algorithm, an efficient dynamic programming

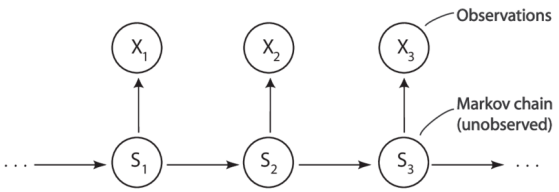

Fig. 7. Basic Structure of HMM

approach for this task. After this, we tuned the Markov chain's transition probabilities given the sequence of observations that we found to create the most likely path. This algorithm, called Baum-Welch, is very similar to the Expectation-Maximization algorithm that we saw in Naive Bayes.

E. LSTM

Recurrent Neural Networks (RNNs) are the typical deep learning architecture for time series applications. Unlike regression predictive modeling, time series also adds the complexity of a sequence dependence among the input variables. The Long Short-Term Memory network (LSTM) network is a type of recurrent neural network used in deep learning because very large architectures can be successfully trained. Using the python libraries Tensorflow and Keras, we developed a model using stacked LSTM layers in order to predict the data for the next timesteps. Below is a representation of the model that was trained for this project.

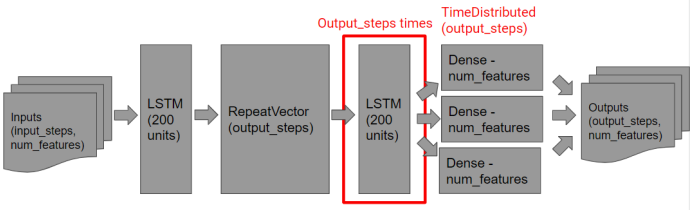

Fig. 8. LSTM Model Architecture

The architecture for the LSTM model consists of a LSTM layer of 200 units connected to a Repeat Vector, which will repeat the input the number of times as the number of output steps. Each of these is then connected to an LSTM layer of 200 hidden units. This is then connected to a Time Distributed layer of Dense networks with the number of units equal to the number of channels in the dataset. Each of these Dense networks predicts the value of the data for each feature at a specific timestep in the future.

F. Generative Adversarial Network

The current strides in understanding for Generative Adversarial Networks has been discussed in the background section. Our attempt was to develop a general architecture to forecast, unlike the common purpose of creating images from noise. Because of their usage in time series problems, we used LSTM layers in both the discriminator and the generator.

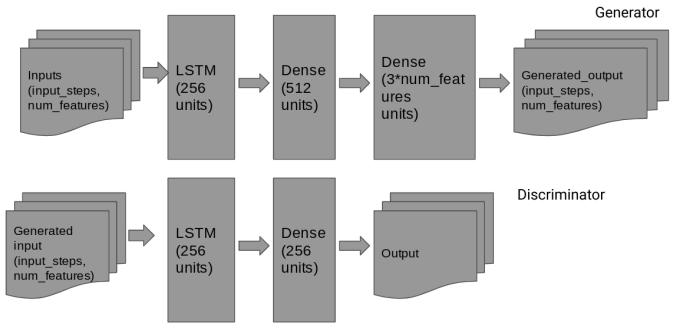

Fig. 9. GAN model Architecture

The generator network was composed of a LSTM layer and 512 unit and 72 unit densely connected layers. The last layer produced the next three predicted points for all channels. The discriminator network was composed of LSTM layer connected to 256 units densely connected layer and output a single number.

IV. Results

TABLE I
SUMMARY OF MODEL FORECASTING ERROR : NMRSD

Model	1-Timestep NMRSD	3-Timestep NMRSD
SVR	0.0043909	0.0068495
SVR+HMM	0.0147577	0.0513297
LSTM	0.024216	0.0028451
GAN	0.20642696	0.22177314

TABLE II
SUMMARY OF MODEL FORECASTING ERROR : MAPE

Model	1-Timestep MAPE	3-Timestep MAPE
SVR	575.56782%	1076.53005%
SVR+HMM	46.94027%	37.91675%
LSTM	666.84268%	950.22060%
GAN	$1.4*10^7$%	$1.5*10^7$%

A. Baseline Analysis

Compared to the baseline model, the only model to see comparable improvement was the LSTM model. Although the models experienced low levels of normalized root mean square deviation, as seen in I. However, surprisingly, the mean average percent error was over 100% for all models that took in large amounts of data, as seen in II. This seems to be because models did not detect a value of zero for data where the value was at or close to zero (the data is sinusoidal in nature, since there are many points in the data where the value nears zero). As a result, the percent error at those points was very high, raising the value of the MAPE. This was solved in the HMM model, where smallest sets of data were used and not as many zero points, but the NMRSD was slightly higher since there was less data to train on.

B. SVR + HMM Analysis

After using the HMM to optimize the SVR approach that sampled 100 points at a time and regressed locally, the MAPE decreased significantly, but the NRMSD remained relatively high.

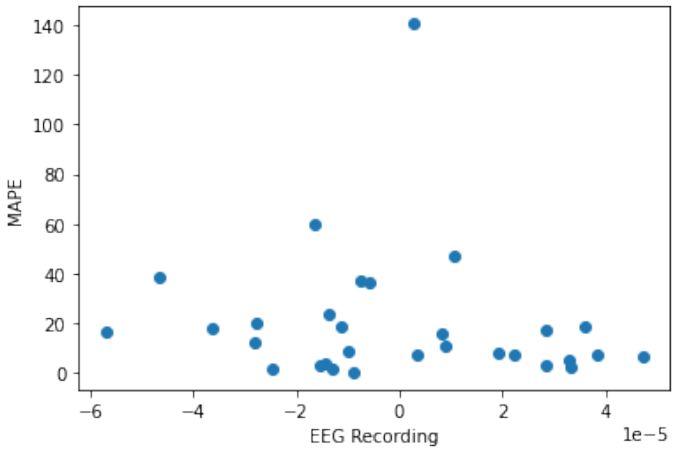

Fig. 10. MAPE correlation with EEG FP1

In Figure 10, we see that the MAPE metric has little or no correlation with the values found at different sampled data in the EEG recording. This means that unline the baseline SVR, the SVR used in coordination with the SVM allowed for better prediction for values that are either very close to or significantly distant from 0.

The NRMSD had very high variability and did not seem like a reliable metric for evaluating the SVRHMM model. We attempted to find a pattern between NRMSD and number of states in the markov model, but there was little correlation with 5 states seemingly being the optimal number. However, the 3 state model was just an outlier in terms of the NRMSD values that the other amounts of states were giving, so in the future, the optimal number of states should be found via gradient descent, along with testing for optimal initial probabilities.

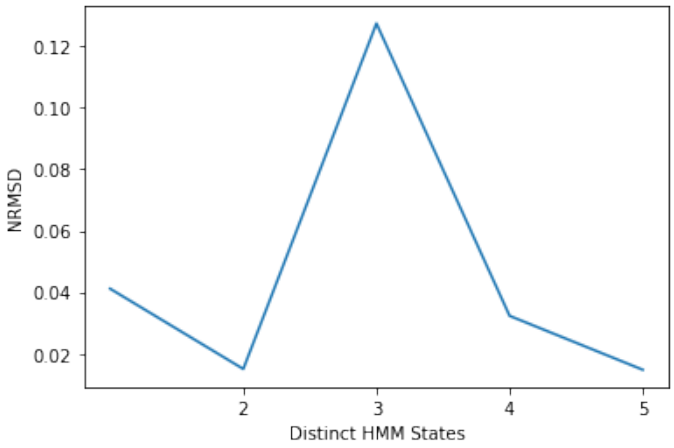

Fig. 11. NRMSD correlation with HMM States for EEG FP1

C. LSTM Analysis

Our LSTM architectures used a regression loss function, specifically, mean absolute error (MAE), with the purpose of making the models more robust against outliers. We trained the model using 50 epochs with a batch size of 128 due to computational hurdles. We also saw a drop in the loss function from increasing the batch size from 72 to 128. Below is a graph of the training loss and the validation loss for each epoch during the training of the model.

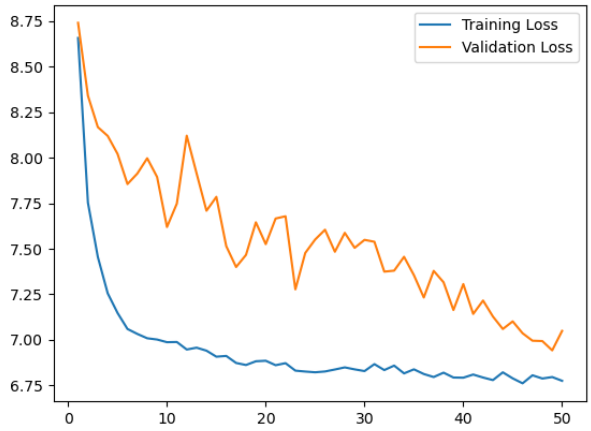

Fig. 12. Single Step Training and Validation Loss

Compared to the baseline model, the LSTM model was an improvement in every metric except for 1-Timestep Mean Average Percent Error, which was roughly 15.8% higher, as seen in II. We believe this is due to the the prediction having a higher percent error at the zero points than the basline model. However, the low NRMSD shows the viability of the model for use in forecasting applications. The baseline model showed a significant difference in performance in prediction for 1-step forecasting versus 3-step forecasting since in the baseline model, the forecast of the data at timestep $t+1$ is dependent on the forecast at timestep t. In contrast, the Time Distributed nature of the LSTM model assured indpendence in forecasting at each time step. As a result, altough it is more computationally intensive it may be a good way to ensure that forecasting error is not compounded at each timestep.

To show the accuracy of the model, we have added a snippet of EEG Data below for a channel that shows the 3-timestep predictions for each timestep in the snippet.

D. GAN Analysis

The GAN model achieved the worst results of all our models. The model's NMRSD was around 100 times worse than other models. We think that it may have been caused by the limiting training time. Our model was trained for only 500 epochs on the subset of the dataset. The training time was limited by our access to cloud GPU.

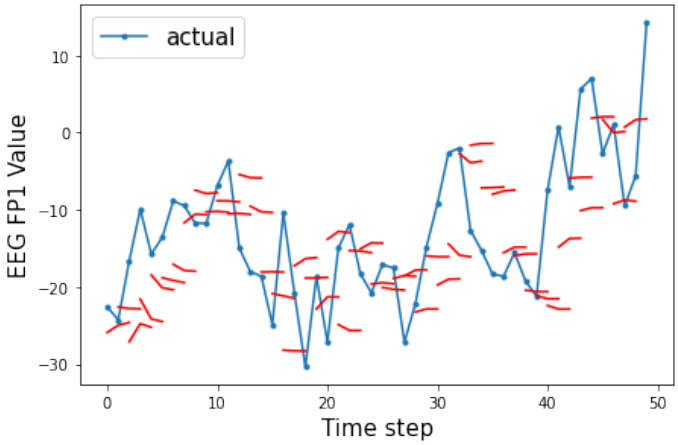

Fig. 13. 3-Timestep LSTM prediction for Data in FP1 Channel

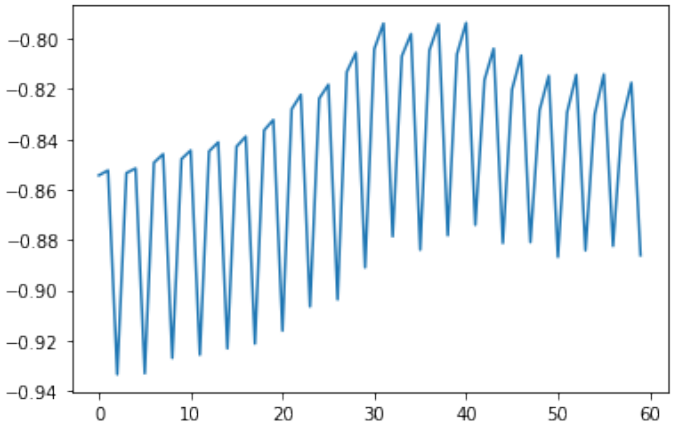

Fig. 14. The GAN output

Another reason may have been mode collapse. The generator outputs were similar for all channels. That discriminator may have learned to always reject them and became locked in a local optima. As shown in Figure 12 the generator have repeated oscillations that are not present in the real data.

V. Conclusion

Through our approach, we tried to show that using generalized data from a group of subjects would not deviate from the actual data of a completely different personal also long as this testing subject did not have any neurological conditions. From the NMRSD of the baseline, Hidden Markov Model, and LSTM models, we have shown that it is feasible to use generalized models in a BCI system without much calibration to a new subject.

Next, we showed that there are tradeoffs between the size of sampling and the fitness of the model, as the SVR + HMM model performed much better considering the Mean Average Percent Error yet had a higher Normalized Root Mean Square Deviation.

One conclusion that we made was that deep learning approaches EEG data forcasting was not significantly better than the baseline as the error showed for the LSTM model. However, for predictions more than one timestep ahead, it is better to use deep approaches because of the independence of prediction at each timestep.

Lastly, due to the failure of the GAN model, we can not conclude whether the GAN can efficiently and accurately predict multiple timesteps from the data. We will explain some ways that the GAN can be changed in order to better fit the data and avoid mode collapse in Future Work. From this, we can also conclude that it is unfeasible to train a GAN model without significant computational power. We also saw that even with the inclusion of models built for time series prediction such as LSTM's, we were not able to generate data that accurately represented the data.

VI. Future work

In the future we plan to improve our GAN model by increasing the training time and modifying the model architecture. Changing the loss function to Wasserstein loss and increasing the weight of the real loss for discriminator could help prevent mode collapse. Together with introduction of R1 regularizer or 0GP regularizer we should be able to achieve better results in the future[16].

Along with changing the loss function, there are many other GAN architectures better suited to time series prediction that we could experiment with, such as the recurrent conditional GAN, as used in [17]. Along with this, the application of GAN's to time series data is a relatively new field and there have been a number of new papers proposing GAN architectures for Time Series data, such as TimeGAN from [18] and TSGAN in [19]. Due to the high number of possible architectures, it is likely that there are many other architectures that we can work to replicate.

Along with changing the GAN, we could also change our approach prediction. In a real-world scenario, in order to upsample noisy and sparse data properly, interpolation may be a better method for producing data with a higher resolution. This may also improve the accuracy of the GAN, since there will be points throughout the snippet of time for which the GAN is generating, which would allow the discriminator to use that deviation to hold a higher set of values, preventing mode collapse.

Along with this, for future work, we may use the forecasting method we produced to upsample the data to create more accuracy on the downstream tasks for a BCI system, such as classification of thoughts and generation of images or words of which the other subject is thinking. From this, we could even extend our view to creating policy-based-apporaches for action based on the signals upsampled using our methods, which could extend to a myriad of applications, from prosthetics to methods of character control in games.

References

[1] Vaadia, Eilon, and Niels Birbaumer. "Grand challenges of brain computer interfaces in the years to come." Frontiers in neuroscience

vol. 3,2 151-4. 15 Sep. 2009, doi:10.3389/neuro.01.015.2009

[2] Pisarchik, Alexander N, et al. “From Novel Technology to Novel Applications: Comment on ‘An Integrated Brain-Machine Interface Platform With Thousands of Channels’ by Elon Musk and Neuralink.” Journal of Medical Internet Research, vol. 21, no. 10, 2019, doi:10.2196/16356.

[3] Mahfuz, Nurshuhada Ismail, Waidah Noh, Nor Jali, Zalisham Abdullah, Dalilah Nordin, Md Jan. (2015). A Classification on Brain Wave Patterns for Parkinson’s Patients Using WEKA. 10.1007/978-3-319-17398-6_3.

[4] Aznan, Nik Khadijah Nik, et al. “Simulating Brain Signals: Creating Synthetic EEG Data via Neural-Based Generative Models for Improved SSVEP Classification.” 2019 International Joint Conference on Neural Networks (IJCNN), 15 Jan. 2019, doi:10.1109/ijcnn.2019.8852227.

[5] Goodfellow, Ian J., et al. “Generative Adversarial Networks.” ArXiv:1406.2661 [Cs, Stat], June 2014.

[6] Luo, Tian-Jian, et al. “EEG Signal Reconstruction Using a Generative Adversarial Network With Wasserstein Distance and Temporal-Spatial-Frequency Loss.” Frontiers in Neuroinformatics, vol. 14, 2020, p. 15.

[7] Svantesson, Mats, et al. “Virtual EEG-Electrodes: Convolutional Neural Networks as a Method for Upsampling or Restoring Channels.” BioRxiv, 2020, pp. BioRxiv, Apr 20, 2020.

[8] K. G. Hartmann, R. T. Schirrmeister, and T. Ball, “EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals,” arXiv preprint arXiv:1806.01875, 2018.

[9] S. Palazzo, C. Spampinato, I. Kavasidis, D. Giordano, and M. Shah, “Generative adversarial networks conditioned by brain signals,” in Int. Conf. Computer Vision, 2017, pp. 3410–3418.

[10] ”MindBigData: the MNIST of Brain Digits”. http://www.mindbigdata.com/opendb/. Accessed 29 July 2020.

[11] Di Fulvio, Tom. “Using Machine Learning to Categorise EEG Signals From The Brain to Words.” Medium, Towards Data Science, 23 Apr. 2019, towardsdatascience.com/using-machine-learning-to-categorise-eeg-signals-from-the-brain-to-words-728aba93b2b3.

[12] Chaves, Alberto Bozal. “Personalized image classification from EEG signals using Deep Learning.” (2017).

[13] Tirupattur, Praveen, et al. “ThoughtViz: : Visualizing Human Thoughts Using Generative Adversarial Network.” 2018 ACM Multimedia Conference on Multimedia Conference - MM ’18, Oct. 2018, pp. 950–958., doi:10.1145/3240508.3240641.

[14] Längkvist, Martin, et al. “A Review of Unsupervised Feature Learning and Deep Learning for Time-Series Modeling.” Pattern Recognition Letters, vol. 42, 1 June 2014, pp. 11–24., doi:10.1016/j.patrec.2014.01.008.

[15] Corley, Isaac A., and Yufei Huang. “Deep EEG Super-Resolution: Upsampling EEG Spatial Resolution with Generative Adversarial Networks.” 2018 IEEE EMBS International Conference on Biomedical Health Informatics (BHI), 9 Apr. 2018, doi:10.1109/bhi.2018.8333379.

[16] Thanh-Tung, Hoang Tran, Truyen Venkatesh, Svetha. (2018). On catastrophic forgetting and mode collapse in Generative Adversarial Networks.

[17] Esteban, Cristóbal, et al. “Real-Valued (Medical) Time Series Generation with Recurrent Conditional GANs.” ArXiv:1706.02633 [Cs, Stat], Dec. 2017. arXiv.org, http://arxiv.org/abs/1706.02633.

[18] Yoon, Jinsung, et al. “Time-Series Generative Adversarial Networks.” Advances in Neural Information Processing Systems 32, edited by H. Wallach et al., Curran Associates, Inc., 2019, pp. 5508–5518. Neural Information Processing Systems, http://papers.nips.cc/paper/8789-time-series-generative-adversarial-networks.pdf.

[19] Smith, Kaleb E., and Anthony O. Smith. “Conditional GAN for Time-series Generation.” ArXiv:2006.16477 [Cs, Stat], June 2020. arXiv.org, http://arxiv.org/abs/2006.16477.