HIGH SCHOOL EDITION
Journal of Student Research

Volume 12 Issue 1 (2023)

Reconstructing Static Memories from the Brain with EEG Feature Extraction and Generative Adversarial Networks

Matthew Zhang[1] and Jeremy Lu[2]

[1]Westlake High School, USA
[2]Saratoga High School, USA

ABSTRACT

Forensic tools and facial recognition techniques have implemented state memory reconstruction in the court of law. Recent research has incorporated machine learning into this field in order to fulfill the rising demand for this service. However, the suggested solutions require expensive cutting-edge equipment such as fMRI and CT scanners. The goal of the present study is to recreate images using user electroencephalography (EEG) input patterns. We combined a discriminator and generator network in order to decrease blur, face distortion, and erroneous features in the ImageNet dataset. An EEG feature matrix was produced using a convolutional neural network (CNN) encoder from spectrogram inputs, which are visual representations of the spectrum of electrical frequencies observed at each electrode. The encoder produced an 85% accuracy when linking spectrograms to the labels of the relevant images. The feature matrix was passed into the GAN during training. The viability of image reconstruction and coherent image representation from the brain were both proved feasible to a dependable degree due to our GAN output images' resemblance to the original dataset. The present work could find applications in future studies as a non-invasive, more affordable option to recreate memories from brain signals.

INTRODUCTION

Recently machine learning applications have explored the development of brain computer interfaces (BCI). Common BCIs such as fMRI or EEG have been used to decode brain activity into simpler forms. Advances in deep learning, such as the application of deep neural networks (DNNs) and CNNs, have allowed for the reconstruction of images based on these forms of brain data [1]. Previously, models have reconstructed visual data based on fMRI datasets. However, fMRI data is not widely accessible, hence finding a large dataset necessary for more complex deep learning models is a difficult task [2]. On the other hand, projecting imagined objects in a visual form in the physical world required the utilization of EEG decoding techniques in many studies. One study adopted a spatial layout to recognize the "texture" of an image and use EEG signals to more accurately reconstruct a perceived image. The study developed a MVAE model, which uses a bimodal variational autoencoder coupled with a binary classifier for the task [7]. LSTM networks identified time-frequency features in spectrograms and achieved a classification accuracy of 90% [3]. In general, the task of classifying images with EEG data seems promising, as studies employing 40 distinct image classes have a classification rate of 50% [5]. Furthermore, static memory reconstruction studies have implemented a variety of architectures—BigBiGAN, auto-encoders, and transformers. In these approaches, the encoding architecture of each model produces latent features, which are then taken to rebuild images, particularly for GAN and autoencoder models [9]. Different approaches to EEG feature extraction include EEG preprocessing and CNNs [8, 9]. Transformer and transfer learning CNNs are also used to produce prominent image features based on EEG input features. The EEG feature extraction process involves band rejection filters to block out line noise, band pass filters to keep features that are necessary, and other smoothing filters [9]. Some studies apply transfer learning models—VGG or ResNet50—

their downsampling encoder architecture contains 2D convolutional layers, batch normalization, and global average pooling, while upsampling or 1D convolutional transpose layers decode these latents [7]. Style loss finds the textural differences in these model-produced images and renders stylistic transitions easier. Main areas for improvement in previous studies include maintaining data diversity, adjusting individual differences in visual processes, reducing the danger of overfitting with smaller data samples, and transitioning from picture to EEG signals [7, 9]. The present study seeks to improve the accuracy of produced images with respect to EEG features with the usage of other machine learning architectures. Specifically for our study, the data includes preset spectrograms that our model inputs.

In order to synthesize various successful architectures, preprocessing algorithms, loss functions, and analytical methods across other related studies, the present work structurally comprises of standard machine learning practices: (1) 10,032 spectrograms of EEG signals; (2) CNN encoder to extract EEG visual features; (3) GAN network including an upsampling generator with latent EEG features as input and a discriminator that determines the textural similarities between original images and generator's depiction of them; (4) binary cross-entropy loss in backpropagation to computationally render features of original images to generated representations; and (5) "Inception Score" metrics for 2000 generated and original dataset images [7-9].

METHODS

The model used in this study contains an encoder network that extracts latent features from the EEG data. The extracted features are then inputted in combination with noise in order to train the generator and discriminator networks in the GAN model. All models ran on an integrated NVIDIA CUDA GPU in Keras (TensorFlow backend). The data was collected by MindBigData's "IMAGENET of the Brain" dataset using five electrodes—AF3, AF4, Pz, T7, and T8—on the Emotiv Insight headset at 128 Hz at three-second intervals [10]. The dataset is made up of 14,012 images (.JPEG) with 10,032 raw EEG files (.csv) with spectrograms (.png) for each file.

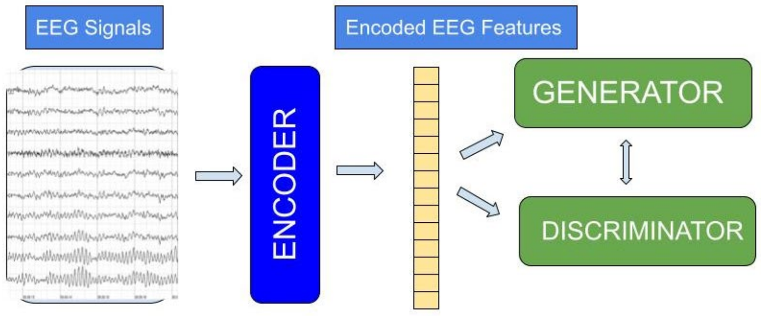

FIG. 1. Overview of the encoder-GAN architecture.

Preprocessing Data

MindBigData provides spectrograms, which are pre-processed as input to the CNN encoder model, for each data trial. Matplotlib, a Python library for plotting visualizations of data, reads the spectrogram images; OpenCV, a library aimed at computer vision and image processing, resizes the images to a uniform shape of 256x256x3 in order to maintain their image qualities. Within the filenames are synset categories, which are appended to an array as labels; Numpy, a library for processing matrices, then converts this array of strings to a one-hot encoded array fit for CNN training.

Encoder Network

However, due to the immense size of the dataset, inserting all data points into the GAN model is not plausible. Therefore, an encoder compresses the EEG data points and extracts important features in a manner that could be put into the GAN model. The encoder receives a spectrogram and outputs a latent matrix that contains only the main features of the EEG data. The encoder developed was a CNN consisting of two blocks of one convolution layer—with the first layer having 32 filters and the other having 64 filters—and one max pooling layer, which is used to extract the maximum value from the previous layer's output, with a pool size of 2x2. The two feedforward dense layers had 128 and 576 units respectively. Furthermore, in order to efficiently extract spectrogram features within the CNN's training process, an encoder model was built with the same input and flatten layer as the output; the new encoder predicts an array of features that would have otherwise been fed forward in the original network. Once the EEG feature extraction is completed, the newly encoded array acts as a fake image input to the GAN network.

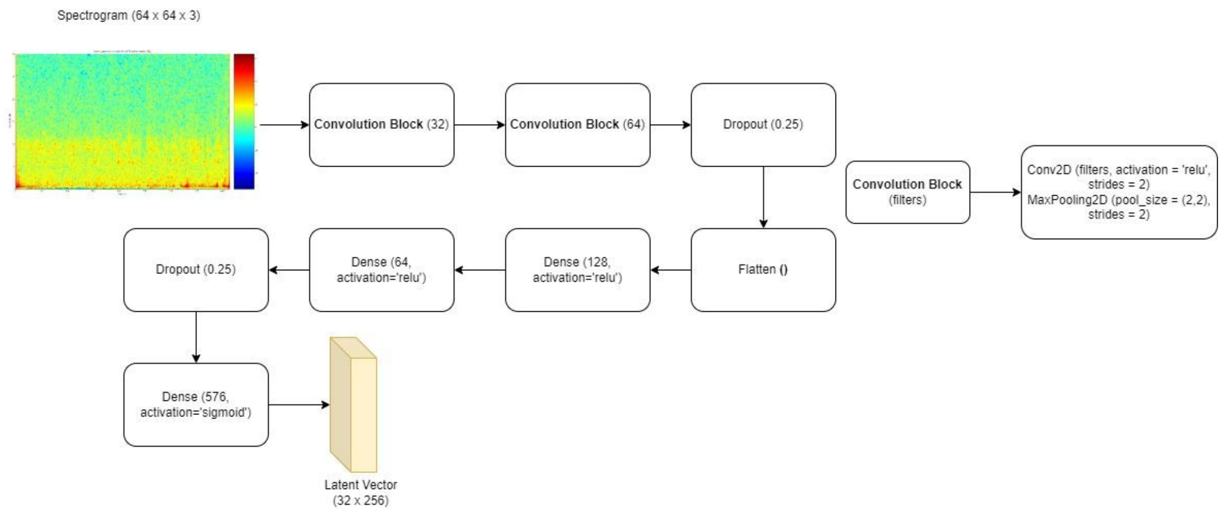
Spectrogram (64 x 64 x 3)
Convolution Block (32)
Convolution Block (64)
Dropout (0.25)
Convolution Block (filters)
Conv2D (filters, activation = 'relu', strides = 2) MaxPooling2D (pool_size = (2,2), strides = 2)
Flatten ()
Dense (128, activation='relu')
Dense (64, activation='relu')
Dropout (0.25)
Dense (576, activation='sigmoid')
Latent Vector (32 x 256)

FIG. 2. Encoder architecture. The spectrogram (64x64x3) is inputted into a set of convolution and max-pooling layers that use kernels to extract maximum data values from the image. The output is a 32x256 latent array that contains EEG features.

In addition, we tested another encoder network consisting of repeat vector, time distributed, and LSTM layers with 100 units, which produced a latent vector of size 388. However, the input shape of the LSTM network did not allow for the inclusion of all EEG data with the essential aspect of dimensionality reduction.

GAN Model

The network type employed to recreate image representations is GAN, which is a machine learning framework that consists of a generator and a discriminator. Preprocessing data for the GAN network requires ImageNet images and EEG feature array batches with a size of 32. Normalizing the images from -1 to 1 proved to add more noise to the images while decreasing the light intensities of the pixels.

$$pixel' = \frac{pixel - 127.5}{127.5}$$

(Image Normalization Formula)

Due to the dataset's diversity in image categories, an MTCNN (Multi-Task Cascaded Convolutional Neural Network) detected faces at a confidence above 90% in each dataset image corresponding to an EEG spectrogram. The chosen images with faces reduced the amount of stylistic variance and, therefore, assisted the GAN in processing specific features of an image, which became a critical characteristic of the GAN's predicted images. To duplicate a probability distribution, GAN implements a loss function that takes into account the separation between the GAN's generated data distribution and the actual data distribution. The loss function is minimized as the generated image resembles the real image more closely, while taking into account the EEG latent space. GAN conducts the update of the generator and discriminator's weights through binary cross entropy.

$$-\frac{1}{N}\sum_{i=1}^{N}(y_i log(p_i) + (1 - y_i) log(1 - p_i))$$

(Binary Cross Entropy Equation)

We implemented a style loss into the GAN training process and updated the GAN optimizer's gradients; however, the images produced by the generator were completely black upon applying the style loss technique. Consequently, the GAN solely ran binary cross entropy as its loss function. Although the Kullback-Leibler divergence loss allowed the GAN to construct colored images, most of the pictures' pixels were noise. The discriminator classifies the real data and fake data from the generator, which in this instance are the fake and real images created by the generator.

The discriminator build consisted of a fully connected network with four convolutional blocks: a convolutional layer, a leaky ReLU activation with an alpha value of 0.2, batch normalization, and a dropout layer with 0.25 probability to prevent overfitting. The final dense layer integrated a sigmoid activation to binary classify real or fake images. The discriminator compiles with the binary cross entropy loss function, Adam optimizer with an initial learning rate of 0.002, beta value of 0.5, and the accuracy metric. In order to maximize the amount of extracted features from the data's unused images, we experimented with training the discriminator on the entire ImageNet dataset with uniform one-hot encoded labels.

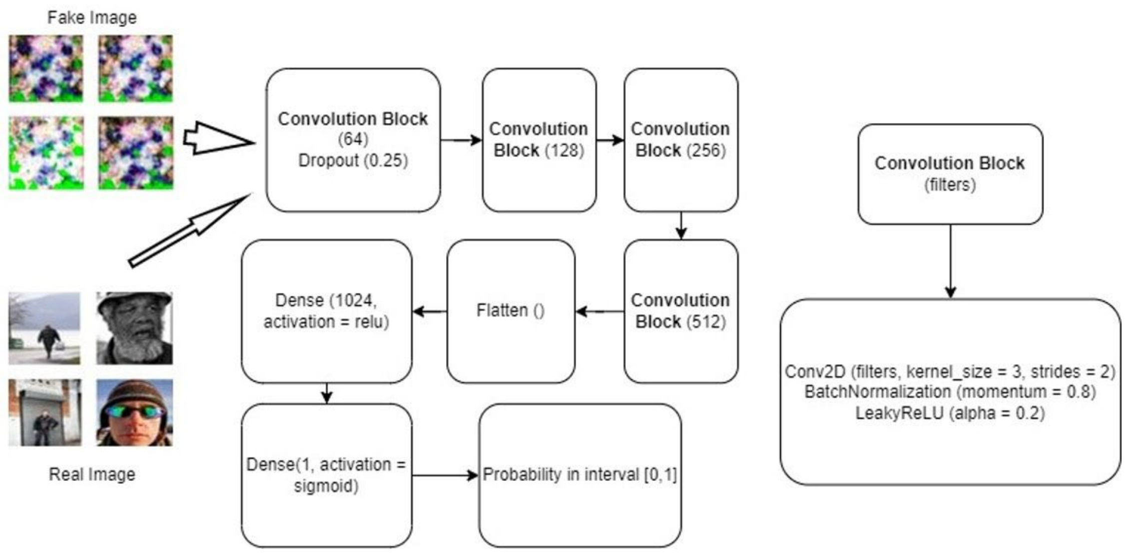
Fake Image
Convolution Block (64) Dropout (0.25)
Convolution Block (128)
Convolution Block (256)
Convolution Block (512)
Flatten ()
Dense (1024, activation = relu)
Dense(1, activation = sigmoid)
Probability in interval [0,1]
Real Image
Convolution Block (filters)
Conv2D (filters, kernel_size = 3, strides = 2)
BatchNormalization (momentum = 0.8)
LeakyReLU (alpha = 0.2)

FIG. 3. Discriminator build. Consists of four convolutional blocks, with each subsequent block having 128, 256, and 512 filters in convolutional layers. Following this is a flatten layer and two dense layers with 128 and 1 cell respectively.

The generator's role in the GAN is to identify specific features that would produce an image most textually similar to the original data over time. Previous studies have laid out the optimal architecture of GAN networks, which have been slightly modified in the present study to cater to the EEG latent inputs and general image reconstruction. The sequential generator model receives the EEG input features of shape 256 concatenated with noise in order to prevent the model from overfitting solely to the EEG inputs. The model architecture included an initial dense layer with 262,144 units, reshaped to fit in the following three deconvolution blocks—which include an upsampling layer, convolutional layer, batch normalization with a momentum of 0.8, and ReLU activation—to increase the dimension size of the features while retaining necessary information. We experimented with replacing the upsampling blocks with convolutional transpose blocks, but this, in turn, decreased the performance of the model. The hyperbolic tangent activation resided in the output layer. The discriminator and generator's blocks initially had 64 filters in their first convolutional layer, which doubled with every following convolutional layer.

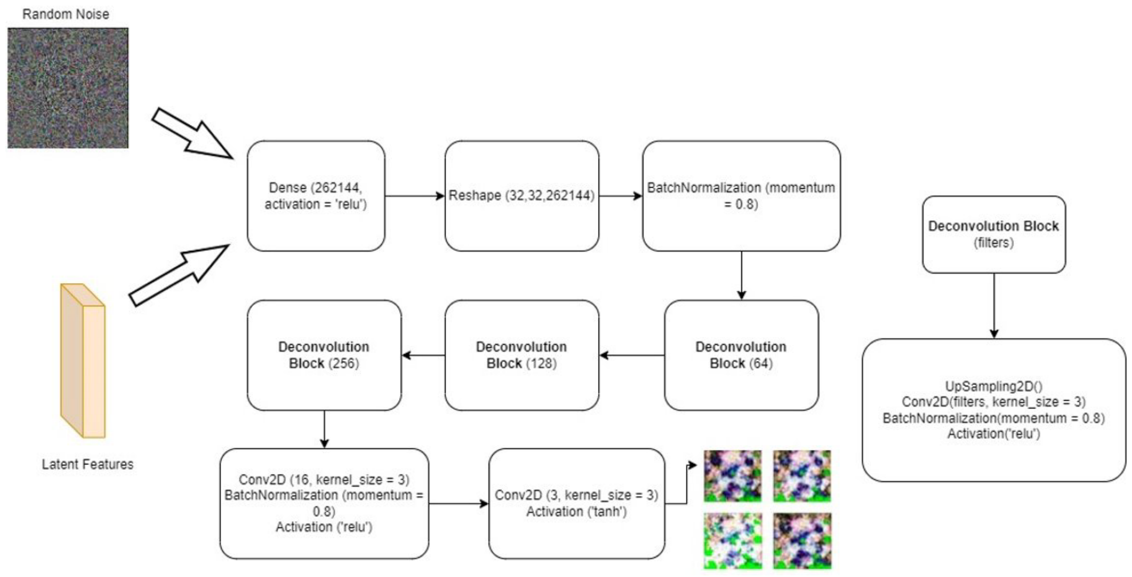
Random Noise
Latent Features
Dense (262144, activation = 'relu')
Reshape (32,32,262144)
BatchNormalization (momentum = 0.8)
Deconvolution Block (64)
Deconvolution Block (128)
Deconvolution Block (256)
Conv2D (16, kernel_size = 3) BatchNormalization (momentum = 0.8) Activation ('relu')
Conv2D (3, kernel_size = 3) Activation ('tanh')
Deconvolution Block (filters)
UpSampling2D() Conv2D(filters, kernel_size = 3) BatchNormalization(momentum = 0.8) Activation('relu')

FIG. 4. Generator build. Consists of three deconvolutional blocks. The first three layers resize features while retaining essential information. The remaining layers translate those features into an image that is put into the discriminator.

The newly created GAN links the discriminator and generator's architecture. The generator's output is put into the discriminator's input. The discriminator should not have the capability to train while the generator produces fake images, so the discriminator's "trainable" parameter is set to false for the GAN's specific architecture. Following the discriminator's own batch training, the generator, which is initialized with random weights, produces images based on inputted EEG features. The generator's weights will then be adjusted based on the discriminator and its own loss. The discriminator and overall GAN network train on batches of 4 for 100,000 epoch iterations.

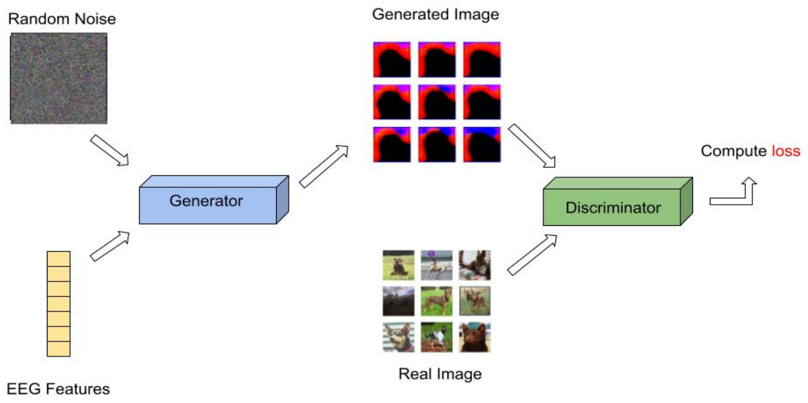

FIG. 5. GAN framework. Noise and an array of latent features are fed into the generator, which produces an image. The discriminator is trained to predict which image is real. The loss is a measure of the accuracy between the generated and real images and is applied by the generator to optimize its weights and biases.

RESULTS

The usage of numerical metrics for coherent GAN-produced images was crucial for evaluating the differences from the generated images' real counterparts. While analyzing image similarities and quality qualitatively was simple, proper assessment of the model's abilities cannot be determined without quantitative analysis. One recent quantitative measure utilized in assessing the outputs of GANs is the Inception Score, which scores groups of images while measuring variety and clarity. If both conditions are satisfied, the score will be high, and vice versa. Calculating this score can be done by inputting the series of images into Keras's Inception network, which in turn will output the Inception score based on the images' resemblance to an object class. For the present study, the Inception Score for 2,000 of the GAN and ImageNet images had a mean of 8.548 and a standard deviation of 1.543. This score demonstrates that the generated images have similar features to their real counterparts, but the GAN's outputs lack diversity and texture.
In addition, we changed the dataset's image shapes from 64x64 to 256x256, which allowed the images to maintain their overall features. The method of increasing the image size allowed the GAN to produce more coherent results utilizing the real images' features. Furthermore, preprocessing methods such as interpolation in resizing prevented blurriness in images, which in turn enhanced the GAN's performance.

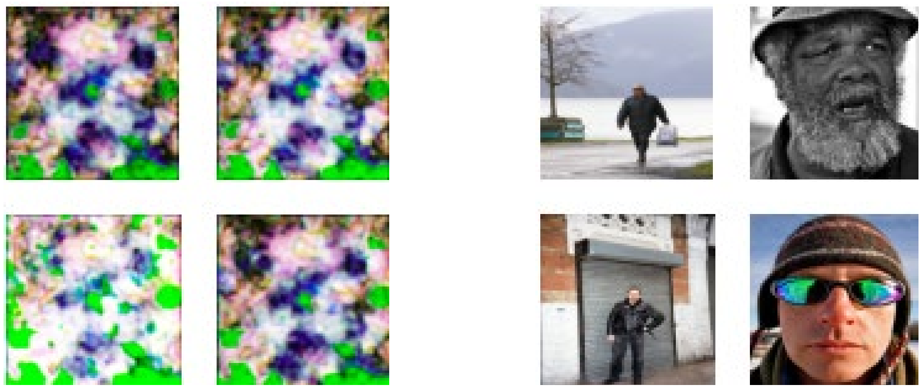

FIG. 6. Generated images from the GAN network (left), and real images from the original dataset (right). Although the images do not look the exact same, the overall features present themselves in the generated images quite clearly.

Hyperparameter tuning also proved to be critical for training the GAN to recognize the minute differences in real and fake images. The learning rate decay caused the learning rate to approach an iteratively smaller value as training went on, which had a negative impact on the GAN's adjustability. Changing the learning rate to a value greater than 0.002 distorted most training images to become overly saturated with solid colors; a learning rate less than the ideal value produced too much noise in the generated samples due to undercorrection of the GAN's weights.

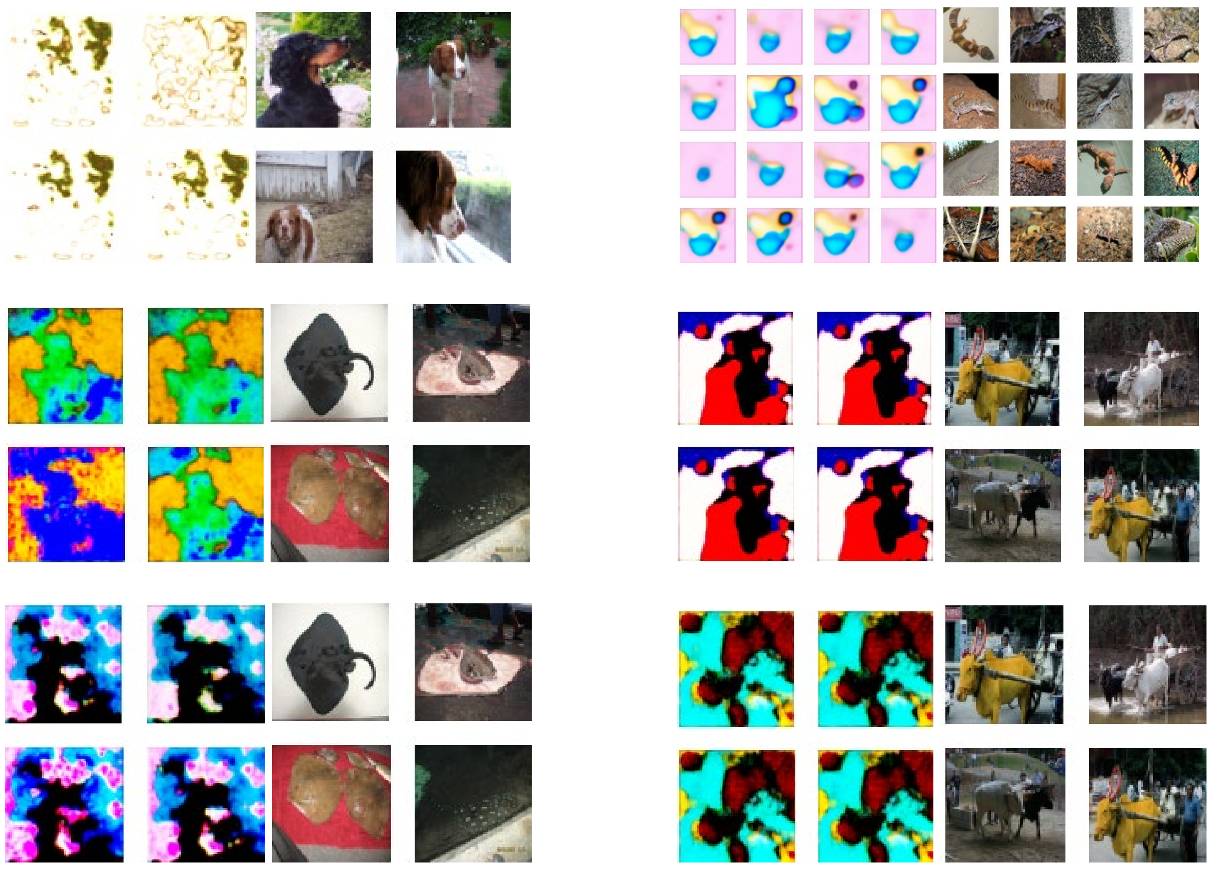

FIG. 7. GAN tested trial samples and their corresponding original images from left to right, top to bottom: 0.02 learning rate; trained on 300 images; inclusion of beta 1 and 2 in the Adam optimizer; trained on the entire dataset; trained on faces at 80% confidence.

The SGD optimizer heavily decreased the encoder's accuracy, while the Adam optimizer brought it up to 85% accuracy in correctly identifying spectrograms with their respective categories.

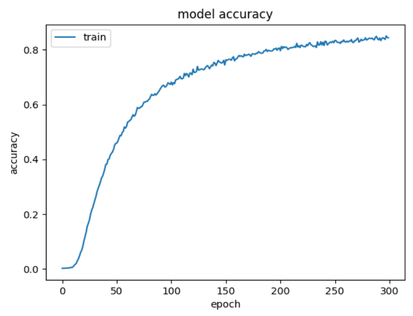

FIG. 8. The CNN encoder accuracy plateaued at an accuracy of 85% over 300 epochs.

Simplifying the encoder's architecture from four convolutional and max pooling layer blocks to two blocks computationally quickened the training process while improving the model from 43% to the current accuracy. On the other hand, reducing layers in the GAN architecture was not possible without either receiving shape errors or severely worsening image quality. For example, replacing upsampling and two-dimensional convolutional layers with a convolutional transpose layer in the generator's deconvolution blocks caused the images to alternate between plain colors: yellow on the first iteration, white on the second, green on the third, etc. Figuring out how to balance alterations in the generator's architecture with the discriminator remained an issue throughout the course of the work. Additionally, completely removing the input noise from the EEG latent space did not cause the model to overfit, but removed the main distinctive features of each generated image.

CONCLUSION

In the current work, we proposed a method for reconstructing images previously seen by a person by training a machine learning algorithm on their recorded EEG frequencies and corresponding images. Because the Inception Score, which was around 8.55 +/- 1.54, for our GAN model was close enough to the score calculations for the GAN model trained on the real CIFAR-10 dataset, which was 8.09 +/- .07, the features appeared in our generated samples. However, the standard deviation of 1.54 conveys a lack of consistency throughout the GAN's images and, therefore, the utilized dataset.

Although the conjoined encoder-GAN model may not have produced the initial intended results, they offer guidance on how future research might enhance the model to produce more realistic images. The design of the encoder involved a CNN model that extracted features from EEG spectrograms. Other architectures—such as LSTM, transfer learning, and transformer models—could be adopted in future trials to extract features from raw EEG data. Another potential encoder fix is the use of principal component analysis (PCA) in order to extract the most principal features and exclude the less important ones. In order to allow the GAN's discriminator to more accurately distinguish patterns between the generator's sample images and the EEG latent feature space, we attempted to concatenate EEG features to the end of the discriminator's output layers. However, due to the immense size of the EEG feature tensor, the dimensions of the EEG features and layer's output were not the same. Further investigation into other methods of EEG feature inclusion into the discriminator may benefit the GAN training process. Moreover, the ImageNet dataset used to train the GAN model contains over 1,000 classes, with approximately 16 images per class. Due to the enormous number of dataset classes as well as the limited amount of training data per label, major issues with the diversity of the image dataset prevented the GAN from pinpointing a category of image to reproduce. For future studies, a dataset with less labels and more training examples per label should be utilized for better results. Additionally, difficulties with the GAN such as mode collapse disabled the GAN's ability to tune images. Mode collapse occurs when the generator only produces a single or small set of outputs. This may happen when the generator finds one type of data that fools the discriminator, which causes the discriminator to have no incentive to change its weights and having the model overfit on that one output. Training the discriminator with the full ImageNet dataset before training the GAN did not solve the issue of mode collapse: the first few generated images had noticeable features, but the proceeding images gradually regressed and did not adjust in quality. We tested less common methods, such as the implementation of a minibatch discrimination layer and the Kullback-Leibler divergence loss, in the GAN but did not implement other advanced solutions such as Wassertein loss due to time constraints. Adjusting the parameters of the Kullback-Leibler divergence loss could be seen as a likely alternative for the GAN; perhaps altering the Adam optimizer's learning rate and beta values would assist in the GAN's ability to modify its weights. Due to the major issues of limited time and computational resources, the present study was unable to identify and determine solutions for issues involved with EEG feature extraction and image reconstruction. By interchanging different machine learning frameworks, optimizing feature extraction models, or addressing the issues presented, future studies can apply more effective procedures from the present study for similar tasks, while avoiding methodologies that hindered the overall model's optimization processes. By exhibiting general features of the original dataset in the GAN's replicated images based on EEG signals, our results demonstrate that image reconstruction from the brain is on the edge of feasibility.

REFERENCES

[1] Shimizu, H., & Srinivasan, R. (2022, January 1). *Improving classification and reconstruction of imagined images from EEG signals*. bioRxiv. Retrieved July 5, 2022, from https://www.biorxiv.org/content/10.1101/2022.06.01.494379v1.full#ref-10

[2] G. Shen, K. Dwivedi, K. Majima, T. Horikawa, and Y. Kamitani, "End-to-End Deep Image Reconstruction From Human Brain Activity," Frontiers in Computational Neuroscience, vol. 13. Frontiers Media SA, Apr. 12, 2019. doi: 10.3389/fncom.2019.00021.

[3] R. Alazrai, A. Al-Saqqaf, F. Al-Hawari, H. Alwanni, and M. I. Daoud, "A Time-Frequency Distribution-Based Approach for Decoding Visually Imagined Objects Using EEG Signals," *IEEE Access*, vol. 8, pp. 138955–138972, 2020, doi: 10.1109/ACCESS.2020.3012918.

[4] P. Bobrov, A. Frolov, C. Cantor, I. Fedulova, M. Bakhnyan, and A. Zhavoronkov, "Brain-Computer Interface Based on Generation of Visual Images," *PLoS ONE*, vol. 6, no. 6, p. e20674, Jun. 2011, doi: 10.1371/journal.pone.0020674.

[5] S. Palazzo, C. Spampinato, J. Schmidt, I. Kavasidis, D. Giordano, and M. Shah, "Correct block-design experiments mitigate temporal correlation bias in EEG classification," *arXiv:2012.03849 [cs, q-bio]*, Nov. 2020, Accessed: Jul. 05, 2022. [Online]. Available: https://arxiv.org/abs/2012.03849

[6] V. Sorin, Y. Barash, E. Konen, and E. Klang, "Creating Artificial Images for Radiology Applications Using Generative Adversarial Networks (GANs) – A Systematic Review," Academic Radiology, vol. 27, no. 8. Elsevier BV, pp. 1175–1185, Aug. 2020. Doi: 10.1016/j.acra.2019.12.024.

[7] S. Wakita, T. Orima, and I. Motoyoshi, "Photorealistic reconstruction of visual texture from EEG signals," *Frontiers*, 01-Jan-1AD. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncom.2021.754587/full. [Accessed: 05-Jul-2022].

[8] T. A. Izzuddin, N. M. Safri, and M. A. Othman, "Compact convolutional neural network (CNN) based on SincNet for end-to-end motor imagery decoding and analysis," Biocybernetics and Biomedical Engineering, vol. 41, no. 4. Elsevier BV, pp. 1629–1645, Oct. 2021. doi: 10.1016/j.bbe.2021.10.001.

[9] K. Ogórek, P. Poryzała, και P. Strumiłło, 'EEG Based Image Reconstruction Using Transformers', Biocybernetics and Biomedical Engineering – Current Trends and Challenges, 2021.

[10] D. Vivancos, *MindBigData "IMAGENET" of The Brain*, 30-Jun-2022. [Online]. Available: http://www.mindbigdata.com/opendb/imagenet.html. [Accessed: 07-Jul-2022].

[11] *Development and validation of an EEG-based real-time emotion recognition system using edge ai computing platform with Convolutional Neural Network System-on-chip design*. IEEE Xplore. (n.d.). Retrieved July 7, 2022, from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8890664&tag=1.