XXIX Congresso de Iniciação Científica UNICAMP 2021

Estudo e Aplicação de Redes Neurais Generativas

Palavras Chave: Modelos Generativos, BCI, *Data Augmentation*

Aluno: Thiago Soares Laitz
Orientador: Prof. Levy Boccato
Faculdade de Engenharia Elétrica e de Computação (FEEC)

1 Introdução

Ao longo dos anos, as redes neurais artificiais vêm demonstrando uma grande versatilidade para tratar problemas associados a inúmeras áreas da ciência. Dentre as possíveis aplicações para os modelos, o aprendizado de representação ganha destaque por viabilizar a extração de atributos relevantes dos dados de entrada, a partir de representações internas no espaço latente. Dentro deste contexto, as redes neurais generativas visam aprender as características típicas dos dados em um domínio de interesse a fim de gerar novas amostras sintéticas que sejam comparáveis às originais [1].

Considerando que com o aumento da quantidade de amostras disponíveis, o efeito de *overfitting* no treinamento de um modelo pode ser reduzido, os novos dados sintéticos podem ser utilizados como uma forma de expandir o banco de dados original, processo que é denominado aumento de dados (DA, do inglês *data augmentation*). Embora todas as aplicações se beneficiem da ampliação do volume de dados, existem problemas, como o projeto de interfaces cérebro-computador (BCIs, do inglês *brain computer interfaces*), em que este processo pode ser ainda mais decisivo, visto que usualmente se percebe uma escassez de dados em virtude da complexidade do processo de coleta e do elevado custo dos equipamentos [2].

Neste trabalho, foram estudados três tipos de redes neurais generativas descritas na literatura: (1) rede auto-codificadora variacional (VAE, do inglês *variational autoencoder*), (2) rede generativa adversária (GAN, do inglês *generative adversarial network*) e (3) rede auto-codificadora adversária (AAE, do inglês *adversarial autoencoder*).

Inicialmente, os modelos foram aplicados no contexto de geração de imagens de dígitos manuscritos no banco de dados MNIST, além do banco de imagens coloridas de quartos de hotéis (LSUN). Por fim, a rede com melhor desempenho foi aplicada ao contexto de BCIs para sintetizar sinais de eletroencefalografia (EEG). As Seções 2 e 3 apresentam os modelos generativos estudados e o funcionamento de uma BCI. A metodologia completa, os resultados obtidos e as conclusões finais do projeto encontram-se nas Seções 4 e 5 deste documento.

2 Modelos Generativos

2.1 VAE

Uma vertente de pesquisa bastante consolidada em aprendizado de representação está associada às redes auto-codificadoras, nas quais o modelo variacional viabiliza a síntese de novas amostras. A estrutura do VAE é semelhante à de um AE (do inglês, *autoencoder*), sendo composta por duas partes: (i) codificador, responsável por transformar as entradas em um vetor de médias e um vetor de variâncias de mesmo tamanho, e (ii) decodificador, responsável por reconstruir a entrada a partir de um vetor de amostras de uma dada distribuição probabilística (e.g, Gaussiana) parametrizada com os vetores disponibilizados pelo codificador [3]. A Figura 1 representa a estrutura do VAE.

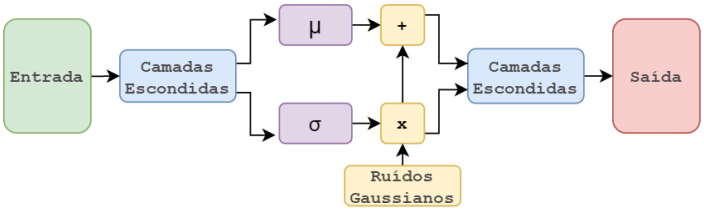

Figura 1: Estrutura geral do VAE.

O treinamento de um VAE considera o objetivo clássico de gerar uma saída tão parecida com a entrada quanto possível, juntamente com um termo especial, chamado de *latent loss*, cuja expressão é dada na Equação (1), sendo $\boldsymbol{\sigma}$ o vetor de desvios padrão e $\boldsymbol{\mu}$ o vetor de média gerados pelo codificador, o qual reflete a intenção de aproximarmos uma determinada distribuição de probabilidades no código interno.

$$\mathcal{L} = -\frac{1}{2}\sum_{i=1}^{K} 1 + log(\sigma_i^2) - \sigma_i^2 - \mu_i^2. \qquad (1)$$

Uma vez finalizado o processo de ajuste dos parâmetros, o VAE é capaz de gerar novos dados que se parecem com aqueles vistos em seu treinamento. Para isso, basta tomar amostras aleatórias para o código interno, de acordo com a distribuição de probabilidades especificada (e.g., Gaussiana), e reconstruir o dado de saída usando o decodificador treinado. Embora ao gerar novos dados a presença do codificador seja desnecessária, é através de sua atuação que a rede decodificadora pode aprender a interpretar a distribuição [4].

O surgimento do VAE motivou o desenvolvimento das redes neurais generativas e, um ano mais tarde, com o surgimento das redes neurais generativas adversárias (GANs, do inglês *generative adversarial networks*) [5], esta linha de pesquisa passou a ser um dos tópicos em destaque dentro de *machine learning*.

2.2 GAN

Diferentemente do VAE, a GAN busca em seu treinamento estabelecer um jogo entre duas redes neurais denominadas de

geradora e discriminadora. A rede geradora tem como função criar novos dados sintéticos a partir de um vetor de amostras aleatórias tomadas de uma PDF alvo (e.g, Gaussiana), enquanto a rede discriminadora busca identificar quais são as amostras reais e quais foram obtidas artificialmente [5]. A Figura 2 representa a estrutura geral do modelo.

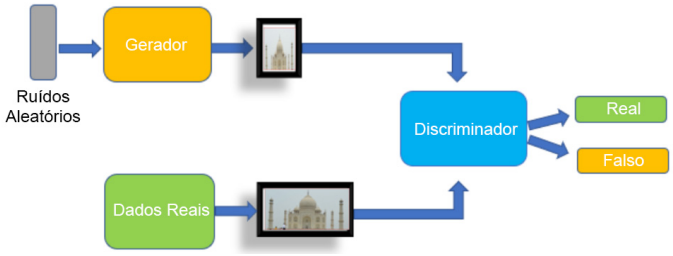

Figura 2: Estrutura geral da GAN.

O processo de treinamento é realizado em duas etapas, sendo a primeira a atualização da rede geradora, na qual os pesos da rede serão atualizados de acordo com as respostas do discriminador. Inicialmente, o gerador cria um *batch* de amostras falsas, o qual é passado à rede discriminadora como se possuísse dados verdadeiros; então, um *batch* de amostras reais também é passado ao discriminador, e a partir das saídas geradas para os dois subconjuntos, calcula-se o vetor gradiente para otimizar a rede geradora.

A segunda etapa de treinamento consiste em alterar os pesos do discriminador. Para isso, a rede geradora novamente cria um *batch* de amostras falsas. Porém, nesse estágio o discriminador busca interpretar essas imagens como falsas, tentando ser capaz de diferenciar o que é sintético do que é real.

Idealmente, o equilíbrio deste jogo se estabelece quando a rede geradora se torna tão hábil em criar dados plausíveis que a rede discriminadora, embora muito boa em reconhecer dados falsos, já não mais consegue perceber a diferença, convergindo para uma taxa de acerto de 50% (equivalente a um classificador aleatório) [5]. A função custo originalmente proposta para o treinamento da GAN é descrita pela equação (2), na qual o discriminador tenta maximizar o seu valor, enquanto o gerador tenta minimizá-la. O termo $log(D(x))$ estima a probabilidade do discriminador classificar um dado (x) real como real, e o termo $log(1 - D(G(z))$ representa a probabilidade de um dado gerado a partir de um ruído aleatório (z) ser classificado como falso.

$$L = E_x[log(D(x))] + E_z[log(1 - D(G(z))] \quad (2)$$

O processo de atualização das redes pode se estagnar caso o discriminador não seja mais capaz de diferenciar o que foi gerado artificialmente do que é real, de forma que a rede geradora não consegue mais melhorar. Além disso, também é possível que a rede geradora aprenda apenas uma única categoria de padrões dentre as várias existentes, o que é denominado de colapso de modo da rede geradora. Com o intuito de evitar ou postergar esses efeitos, existem técnicas que podem auxiliar o seu treinamento [6].

Uma das técnicas, conhecida como *minibatch discrimination*, busca postergar o colapso de modo do gerador. Para isso, é criada uma projeção em matriz de vários atributos extraídos, de forma que a rede discriminadora analise várias amostras ao mesmo tempo em vez de uma amostra isolada. Outra possibilidade é a adição de um termo extra na função custo com base no valor médio dos pesos sinápticos a fim de evitar grandes oscilações ao longo das épocas. Este processo é conhecido como *historical averaging*.

Com relação à classificação do discriminador, considerando que as classes utilizam o valor 0 e 1 como saída, é possível alterar o valor de 1 para 0,9, com o intuito de auxiliar o modelo a se mover na direção da distribuição dos dados. Esta técnica é chamada de *one-sided label smoothing*. Por fim, o *virtual batch normalization* busca realizar a normalização dos dados com base em estatísticas coletadas de um *batch* de referência, visto que em uma camada de *batch normalization* comum, a normalização será muito dependente de outras entradas, o que pode levar a uma saída uniformizada da rede.

2.3 AAE

Analogamente ao VAE, o AAE busca em seu treinamento ser capaz de a partir de uma distribuição, gerar novas amostras condizentes com os dados reais. No entanto, ao invés de utilizar a divergência de Kullback-Leibler para o casamento da PDF interna e a PDF alvo, esse processo é realizado com base na noção de aprendizado adversário. O AAE é constituído por três redes neurais: (i) codificador, cuja estrutura busca representar os dados de entrada em uma determinada PDF (ii) discriminador, responsável por tentar determinar quais amostras (no espaço latente) são verdadeiras e quais foram geradas pelo codificador e (iii) decodificador, utilizado para reconstruir os dados de entrada a partir do vetor latente [7]. A estrutura é ilustrada na Figura 3.

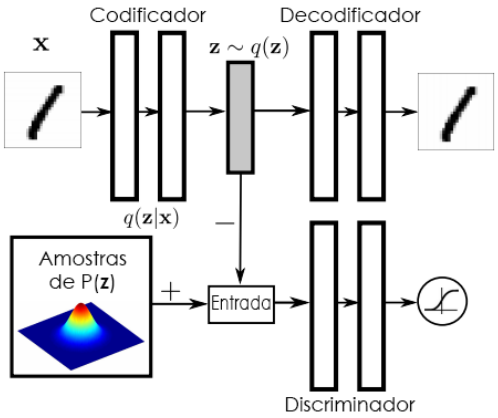

Figura 3: Estrutura de um AAE. Adaptado de [7]

A atualização de parâmetros é realizada em duas etapas executadas para cada *minibatch*: (i) reconstrução e (ii) regularização. Inicialmente, na reconstrução apenas os pesos da rede auto-codificadora são atualizados para minimizar o erro de reconstrução na saída. Na regularização, inicialmente é atualizado o discriminador visando diferenciar qual amostra é verdadeira e qual é falsa. Por fim, os pesos do codificador são melhorados para confundir o discriminador. Após o treinamento, podemos utilizar apenas a rede decodificadora com um vetor de amostra da PDF alvo como entrada para gerar novos dados sintéticos.

3 Interfaces cérebro-computador

Em termos simples, um sistema BCI corresponde a um dispositivo capaz de fornecer ao seu usuário um canal de comunicação considerado não-muscular, com base no monitoramento da atividade cerebral deste indivíduo e na identificação de sua intenção de realizar determinada tarefa [8]. Dentre os vários paradigmas existentes para uma BCI, a abordagem baseada em potenciais visualmente evocados em regime estacionário (SSVEP, do inglês *steady-state visually evoked potentials*) explora o fato de que quando o usuário é

estimulado visualmente, seja por um LED ou por projeções em uma tela de computador, é possível observar um aumento de energia nos sinais cerebrais – tipicamente registrados via EEG – em torno da frequência relacionada ao estímulo e também em suas harmônicas. Sendo assim, uma BCI-SSVEP processa os sinais de EEG, com a finalidade de reconhecer a frequência de estimulação correspondente ao padrão visual selecionado pelo usuário, o qual identifica um comando específico na aplicação alvo. Trata-se, portanto, de um problema de classificação cujos dados de entrada são registros de EEG de vários eletrodos coletados em um determinado intervalo de tempo.

O processamento dos sinais cerebrais em uma BCI-SSVEP comumente envolve os seguintes estágios, conforme ilustrado na Figura 4: (1) pré-filtragem, para a remoção dos artefatos que persistiram na base de dados; (2) extração de atributos, de modo a ter uma descrição compacta e representativa das informações mais relevantes dos sinais cerebrais observados; (3) seleção de atributos; e, finalmente, (4) classificação, onde se identifica o comando desejado pelo usuário a partir do processamento dos atributos extraídos [9].

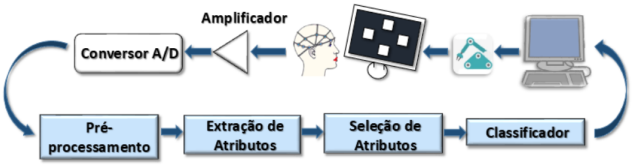

Figura 4: Diagrama com todas as etapas de uma BCI. Adaptado de [10].

Dada a limitação de dados de EEG disponíveis e o alto custo em se preparar um experimento para sua coleta, o emprego de modelos generativos para imitar o padrão de atividade cerebral de usuários e, assim, criar novos registros de EEG pode ser relevante para o projeto de BCIs. A seguir, os resultados encontrados e a metodologia utilizada são explicados.

4 Metodologia e Resultados

4.1 MNIST

Inicialmente, a proposta do trabalho foi avaliar o comportamento dos modelos generativos no contexto do famoso banco de dados de dígitos manuscritos (MNIST), que possui 60 mil imagens com 28×28 pixels para treinamento, e 10 mil para validação. Para a construção das redes neurais foi utilizada a linguagem Python e as bibliotecas *Tensorflow* e *Keras*.

4.1.1 VAE

A estrutura da rede neural do VAE foi construída de forma simétrica, na qual o codificador possui um vetor de entrada com 784 valores normalizados de [0,1] e duas camadas densas de 150 e 100 neurônios com função de ativação *selu*. Para o decodificador, foram utilizadas duas camadas densas de 100 e 150 neurônios com função *selu*, além de uma camada de saída com 784 neurônios (redimensionadas posteriormente para 28×28) e função de ativação *sigmoid*. A representação interna, amostrada a partir de um vetor com 2 elementos de média e variância, é gerada pela segunda camada do codificador e funciona como entrada para o decodificador.

O treinamento foi realizado por 1000 épocas utilizando como otimizador o *RMSprop*, sendo a função custo de reconstrução dada pela entropia cruzada binária, além de um termo representando a divergência de Kullback-Leibler. Uma vez finalizado o treinamento, novas imagens podem ser sintetizadas pelo decodificador através da inserção de vetores aleatórios tomados a partir da PDF alvo.

4.1.2 GAN

Para o banco de dados MNIST, as amostras foram normalizadas no intervalo [-1,1]. A estrutura da rede é descrita na Figura 5, sendo que as camadas de convolução transposta no gerador possuem função de ativação selu, exceto a última, que usa a função tangente hiperbólica. Para o discriminador, as camadas convolucionais possuem função de ativação *leaky ReLU(0.2)* e saída densa com função *sigmoid*. O treinamento foi executado por 100 épocas.

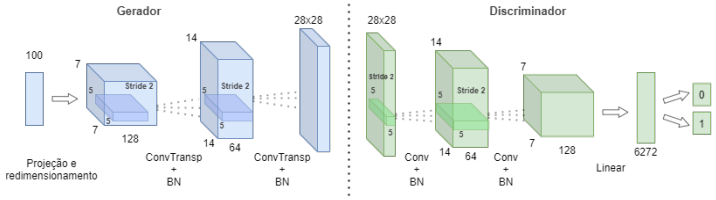

Figura 5: Estrutura da GAN para o banco de dados MNIST.

Os dados foram gerados pela GAN através da rede geradora, considerando que em sua entrada temos um vetor com 100 elementos gerados por uma distribuição Gaussiana padrão.

4.1.3 AAE

Para o AAE, a rede codificadora possui 3 camadas, sendo as duas primeiras densas de 1000 neurônios com *relu* e saída de 8 neurônios sem função de ativação. A rede decodificadora, semelhantemente, possui as mesmas duas camadas densas de 1000 neurônios, com saída de 784 neurônios e ativação *sigmoid*. Por fim, o discriminador possui duas camadas com 1000 neurônios, além de uma saída única com função *sigmoid*. O treinamento foi executado por 3000 épocas. Os dados sintéticos foram gerados a partir da rede decodificadora.

4.1.4 Comparação

De modo geral, podemos perceber que os modelos baseados em redes auto-codificadoras tendem a gerar imagens um pouco mais ruidosas, com menor nitidez nos traços característicos dos dígitos. Por outro lado, a GAN consegue gerar traços com maior definição, mas também um pouco distorcidos. Na Figura 6 são apresentados alguns resultados sintetizados pelos modelos.

Figura 6: Comparação de modelos: MNIST.

4.2 LSUN:Bedroom

Para esta base de imagens coloridas, selecionamos aleatoriamente apenas 5% das amostras redimensionadas para 64×64 pixels, totalizando 154841 imagens, e aplicamos apenas os dois modelos que tiveram o melhor desempenho no MNIST: GAN e AAE.

4.2.1 GAN

Para o LSUN:Bedroom, as amostras foram normalizadas no intervalo [-1,1]. Tanto o codificador quanto o decodificador são formados por camadas de convolução transposta com função de ativação relu, exceto a camada de saída, que usa tangente hiperbólica. Para o discriminador, as camadas

utilizam a função *leaky ReLU(0.2)* e saída densa com função *sigmoid*, além de utilizar a técnica do *minibatch discrimination*. O treinamento foi executado por 5 épocas.

4.2.2 AAE

Para o LSUN:Bedroom, as amostras foram normalizadas no intervalo [-1,1]. A estrutura do codificador e decodificador são compostas por camadas convolucionais transpostas com função *relu* e tangente hiperbólica para o gerador. Para o discriminador, utilizamos duas camadas densas com 1000 neurônios e função *leaky ReLU(0.2)*. O treinamento foi executado por 5 épocas.

4.2.3 Comparação

A partir da Figura 7 concluimos que o modelo GAN, embora treinado com apenas 5% do *dataset* completo, consegue produzir imagens condizentes com os dados originais. Por outro lado, o AAE não foi capaz de aprender as características das imagens de quartos de hotéis, gerando padrões inconsistentes.

Figura 7: Comparação de modelos: LSUN.

4.3 BCI

Para a síntese de sinais de EEG, utilizamos apenas o modelo que se mostrou mais competente em criar padrões verossímeis de imagens para as bases MNIST e LSUN: a GAN. Inicialmente, foi avaliado o comportamento da BCI classificando e gerando os dados apenas no domínio do tempo. Por fim, avaliamos os dados no domínio da frequência.

Para a avaliação da qualidade dos dados gerados é necessária a utilização de classificadores nas frequências utilizadas: 12 e 15 Hz. Para isso, o experimentou contou com a participação de 4 classificadores diferentes: (1) SVM (*support vector machine*) polinomial, (2) SVM linear, (3) SVM gaussiano e, por fim, (4) uma estrutura de rede neural descrita por [11] e representada na Figura 8, que consiste de camadas densas combinadas com camadas de convolução 1D usando a função de ativação *ReLU*, além de uma saída com dois neurônios e função *softmax*.

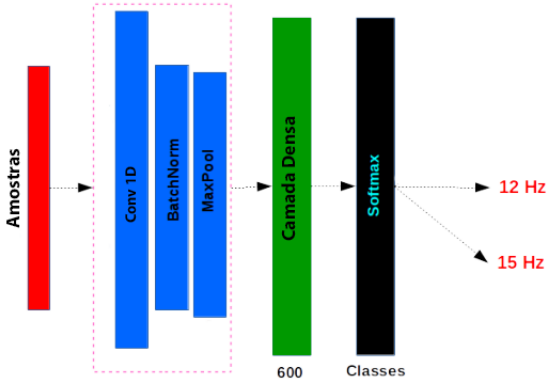

Figura 8: Estrutura do classificador neural. Adaptado de [11].

banco de dados já disponível no grupo de pesquisa (CAAE 0617.0.146.000-10) consiste de amostras de 32 pessoas ao longo de 8 sessões, avaliando 4 frequências (6 Hz, 10 Hz, 12 Hz e 15 Hz) por 12 s, com uma taxa de amostragem de 256 Hz. Nos experimentos, consideramos somente os registros do eletrodo Oz para os dez primeiros indivíduos. Os dados foram divididos em janelas de 3 s, com sobreposição de 2 s, de modo que temos 60 padrões para treinamento e 20 de validação por frequência e por indivíduo.

Para os dados sintéticos, a estrutura da GAN foi construída baseada em camadas convolucionais 1D transpostas para a rede geradora e convolucionais 1D para a rede discriminadora. O gerador recebe como entrada 64 amostras retiradas de uma distribuição gaussiana com desvio padrão igual a 3.

4.3.1 GAN e BCI no domínio do tempo

Inicialmente, é realizado o treinamento dos classificadores 5 vezes sem a presença de dados sintéticos por 100 épocas para a rede neural e 200 vezes variando parâmetros para os SVMs e considera-se como resultado a melhor precisão obtida dentre todas as épocas de todos os treinamentos. Após o treinamento sem DA, treina-se novamente os classificadores agora com a inserção de dados considerando: (1) 50 dados por frequência, (2) 100 dados por frequência e (3) 150 dados por frequência. A tabela 1 mostra os resultados obtidos, com destaque para o DA aplicado na rede neural CNN (do inglês, *convolutional neural network*) que obteve um aumento de precisão em 9 dos 10 casos aplicados. Os melhores desempenhos para cada classificador se encontram em negrito. A Figura 9 mostra um dado sintético em comparação com um dado real.

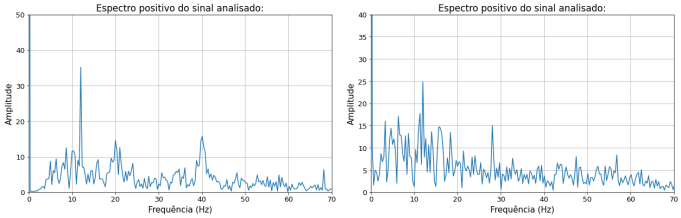

Figura 9: À esquerda o espectro de um dado real, à direita o espectro de um dado sintético com a GAN treinada no tempo.

4.3.2 GAN e BCI no domínio da frequência

Semelhantemente à análise temporal, a estrutura utilizada é idêntica, com pequenos ajustes nas camadas convolucionais para processar/gerar as 384 amostras da transformada de Fourier. Novamente, os classificadores são treinados sem e com DA para comparar os resultados. A tabela 2 mostra o desempenho obtido com destaque para a rede CNN que em alguns casos trouxe ganhos superiores a 10%. É válido enfatizar que o paradigma SSVEP possui um apelo natural ao domínio da frequência, o que foi confirmado pela maior precisão do classificador e pela qualidade dos espectros diretamente gerados pela GAN. A figura 10 mostra a comparação normalizada de um espectro gerado e um espectro real.

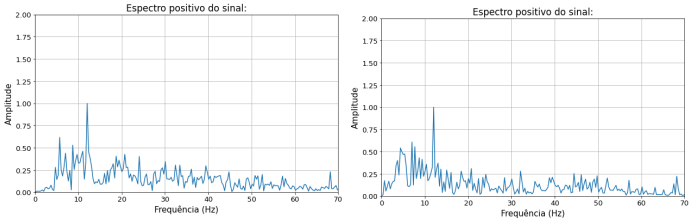

Figura 10: À esquerda o espectro de um dado real, à direita o espectro de um dado sintético com a GAN treinada em frequência.

4.4 Comparação

A partir das tabelas 1 e 2 pode-se observar que o desempenho de BCIs em frequência tende a ser mais preciso, ao mesmo

Tabela 1: Desempenho dos classificadores com dados temporais.

Suj.	SVM Poly	SVM Linear	SVM Gau.	CNN	Poly + 50	Poly + 100	Poly + 150	Lin. + 50	Lin. + 100	Lin. + 150	Gau. + 50	Gau. + 100	Gau. + 150	CNN + 50	CNN + 100	CNN + 150
0	0.6000	0.5000	0.5750	0.600	0.6000	0.6000	0.6250	0.4750	0.4750	0.4500	0.6000	0.6250	0.6000	0.550	**0.650**	0.625
1	0.6500	0.6000	0.6250	0.675	0.6500	0.7000	0.7000	0.6750	0.6750	0.6500	0.6250	0.6250	0.6250	0.675	**0.725**	0.700
2	0.6000	0.4500	0.5500	0.575	0.6000	0.5750	0.6000	0.4500	0.4500	0.4750	0.6000	0.6000	0.6000	0.575	**0.650**	0.650
3	0.6750	0.6000	0.6750	0.600	0.6500	0.6500	0.5250	0.6250	0.6750	0.7000	0.6750	0.6500	0.6750	0.575	0.575	**0.725**
4	0.6000	0.5000	0.5000	0.625	0.6000	0.5750	0.5750	0.6000	0.6000	0.5500	0.6000	0.6000	0.5750	**0.675**	0.625	0.575
5	0.7250	0.6500	0.7250	0.700	0.7000	0.7000	0.7000	0.6500	0.6500	0.6500	0.6750	0.6500	0.6500	0.600	0.625	**0.750**
6	0.7000	0.7250	0.6000	0.700	0.7000	0.6750	0.6500	0.7750	0.7500	0.7000	0.6000	0.5250	0.5500	0.750	**0.775**	0.750
7	0.9750	0.9750	0.9250	**0.975**	0.9750	0.9750	0.9750	0.9750	0.9500	0.9500	0.9500	0.9750	0.9250	0.975	0.975	0.975
8	0.5500	0.5000	0.4500	0.525	**0.6500**	0.6000	0.6250	0.4750	0.4750	0.4500	0.5000	0.5250	0.5000	0.500	0.600	0.525
9	0.5750	0.4500	0.6000	0.525	0.6250	0.6250	0.6250	0.4250	0.4500	0.4500	0.6000	0.6250	**0.6500**	0.525	0.550	0.550

Tabela 2: Desempenho dos classificadores com dados em frequência.

Suj.	SVM Poly	SVM Linear	SVM Gau.	CNN	Poly + 50	Poly + 100	Poly + 150	Lin. + 50	Lin. + 100	Lin. + 150	Gau. + 50	Gau. + 100	Gau. + 150	CNN + 50	CNN + 100	CNN + 150
0	0.9000	0.9000	0.8750	1.0000	0.8500	0.8500	0.8250	0.9000	0.8750	0.8500	0.8500	0.7750	0.7000	**1.0000**	1.0000	1.0000
1	0.5750	0.6250	0.5250	0.6750	0.5750	0.5750	0.5000	0.5750	0.5500	0.5250	0.5250	0.5250	0.5250	0.6500	**0.7250**	0.7000
2	0.6250	0.6500	0.6250	0.7750	0.6250	0.6000	0.6250	0.6500	0.6250	0.6000	0.6000	0.6000	0.6000	**0.8750**	0.7750	0.7750
3	0.5750	0.5750	0.5500	**0.6500**	0.5000	0.5250	0.5500	0.5000	0.4500	0.4500	0.5250	0.5250	0.5250	0.5250	0.6500	0.6250
4	0.5250	0.5000	0.4750	0.5750	0.5000	0.5250	0.5250	0.5000	0.4500	0.4750	0.5750	0.5250	0.5250	0.5000	0.5750	**0.6500**
5	0.7750	0.9250	0.8500	0.9750	0.8250	0.8000	0.8250	0.9750	0.9500	0.9500	0.8500	0.8250	0.7500	**1.0000**	1.0000	1.0000
6	0.6250	0.7750	0.6250	**0.8250**	0.6750	0.6250	0.7250	0.7000	0.6750	0.7000	0.5750	0.6500	0.5750	0.7750	0.8000	0.8000
7	0.8750	0.9250	0.8750	0.9500	0.8750	0.8500	0.8500	0.9250	0.9250	0.9250	0.8250	0.8000	0.8250	0.9500	0.9500	**1.0000**
8	0.5500	0.4750	0.5000	**0.6000**	0.5250	0.5250	0.5250	0.5000	0.5250	0.5250	0.5250	0.5250	0.5000	0.5250	0.5750	0.5750
9	0.9750	0.9750	0.9750	1.0000	0.9750	0.9750	0.9500	0.9750	0.9750	0.9750	0.9750	0.9500	0.9250	**1.0000**	1.0000	1.0000

tempo que para modelos generativos no âmbito de BCIs SSVEP treinadas em frequência, os dados gerados possuem uma qualidade superior em relação ao dado bruto (temporal), trazendo ganhos que em alguns sujeitos ultrapassam a marca de 10%. Não obstante, quase não foi possível perceber ganhos com DA em frequência no caso de SVM.

5 Conclusões

Este trabalho de iniciação científica pôde mostrar a capacidade dos modelos generativos em extrair atributos relevantes dos bancos de dados a fim de gerar dados sintéticos condizentes. No âmbito de BCIs SSVEP, eles devem ser considerados como uma ferramenta importante no aumento de precisão.

Referências

[1] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 2nd ed. O'Reilly Media, 2019.

[2] E. Lashgari, D. Liang, and U. Maoz, "Data augmentation for deep-learning-based electroencephalography," *Journal of Neuroscience Methods*, p. 108885, 2020.

[3] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv: 1312.6114*, 2013.

[4] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.

[6] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *arXiv preprint arXiv:1606.03498*, 2016.

[7] A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow, "Adversarial autoencoders," *CoRR*, vol. abs/1511.05644, 2015. [Online]. Available: http://arxiv.org/abs/1511.05644

[8] J. Wolpaw and E. W. Wolpaw, *Brain-Computer Interfaces: Principles and Practice*. Oxford University Press, 2012.

[9] B. Graimann, G. Pfurtscheller, and B. Allison, *Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction*. Springer, 2010.

[10] S. N. d. C. LEITE, "Contribuições ao desenvolvimento de interfaces cérebro-computador baseadas em potenciais evocados visualmente em regime estacionário," Ph.D. dissertation, Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, 2016.

[11] N. K. N. Aznan, A. Atapour-Abarghouei, S. Bonner, J. D. Connolly, N. Al Moubayed, and T. P. Breckon, "Simulating brain signals: Creating synthetic EEG data via neural-based generative models for improved SSVEP classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.