Predição Contrafactual de ECG's para Análise de Prognósticos Cardíacos

Eduardo Diniz Mio Anisio Mendes Lacerda Wagner Meira Jr
Departamento de Ciência da Computação
Instituto de Ciências Exatas
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais 31270-901
Email: {eduardomio,anisio,meira}@dcc.ufmg.br

***Abstract*—As doenças cardiovasculares representam um grande desafio para os sistemas de saúde devido às complicações crônicas e à alta taxa de mortalidade. Este projeto propõe uma técnica para prever a evolução do paciente ao longo do tempo para evitar danos a sua saúde e poder compor tratamentos que melhor se adequem a sua situação. O projeto é composto por quatro fases: um extrator de atributos, um mapa causal para relacionar as informações do exame e do paciente, um modelo geracional para produção de contrafactuais ao longo do tempo como prognóstico futuro, e a geração de uma métrica de saúde do paciente. A primeira parte do projeto, apresentada neste relatório, discute as duas primeiras fases e apresenta os resultados do extrator de atributos e do mapa causal.**

I. Introdução

As doenças cardiovasculares são a principal causa de morte globalmente [1], com o eletrocardiograma (ECG) sendo essencial para diagnósticos. A digitalização dos ECGs destacou a análise computadorizada, embora os algoritmos tradicionais sejam limitados, servindo apenas como suporte diagnóstico [2]. Avanços em aprendizado de máquina e inferência causal, exemplificados pelo uso de geradores contrafactuais [3] propõem melhorias na análise de ECG, aplicando técnicas contrafactuais para modelar o impacto de intervenções e prever condições futuras dos pacientes, baseando-se na metodologia de Judea Pearl [4], que visa aprimorar a precisão preditiva ao evitar correlações espúrias [5], utilizando mapas causais para selecionar atributos relevantes. A metodologia foca na geração de cenários contrafactuais para ilustrar o comportamento dos sinais de ECG sob variadas condições, validada pelo banco de dados CODE-15 [6], demonstrando o potencial das técnicas contrafactuais em representar diferentes cenários de intervenção e antecipar a condição cardíaca futura do paciente.

A. Objetivo

O objetivo deste projeto é pesquisar, implementar e avaliar modelos e algoritmos capazes de prever a evolução da saúde cardíaca do paciente ao longo do tempo. Para realizar a tarefa, será empregada uma técnica contrafactual a partir de uma rede neuronal [3], permitindo a análise de como as intervenções ou alterações específicas podem afetar a trajetória da saúde cardíaca. Esta abordagem visa não apenas prever mudanças na saúde do paciente, mas também entender as relações causais subjacentes que direcionam tais evoluções. A avaliação do projeto pode ser feita utilizando um grupo controle para testar o valor preditivo do projeto, pois existem pacientes com múltiplos exames na base de dados, que podem ser utilizados para testar o valor preditivo da técnica.

II. Trabalhos Correlatos

[5] investigam o impacto de correlações espúrias na detecção de distribuições fora do conjunto de dados de treinamento, discussão fundamental para a área pois mesmo em diferentes hospitais na mesma região a distribuição de pacientes se altera o suficiente para impactar a performance de modelos de IA [7]. [3] propõem modelos causais estruturais profundos para inferência contrafactual computacionalmente tratável, que propõem ideias fundamentais para o projeto pois inferência em distribuições arbitrárias tem complexidade computacional cúbica em tempo e quadrática em espaço [8]. Finalmente, [9] discutem o DAGMA, um método para aprender grafos acíclicos direcionados (DAGs) através de matrizes M e uma caracterização da aciclicidade via determinante logarítmico, que compõe a vantagem de ser computacionalmente eficiente e ter a flexibilidade para a descoberta de padrões não lineares.

III. Metodologia

Explorando as fases do projeto como um todo, temos quatro principais. A primeira constituiu a análise e preparação do conjunto de dados existente. Também foi implementado um extrator de atributos, cuja validade é assegurada por meio da análise do erro de reconstrução do modelo. O mapa causal pode ser considerado como uma proposta de experimento, e por isso, foram testados vários paradigmas de desenvolvimento e por fim foi escolhido o melhor. As etapas que serão feitas no próximo semestre serão o desenvolvimento de um modelo para geração dos dados utilizando o mapa causal e um avaliador da saúde do paciente baseado no espaço latente produzido na primeira etapa. Logo, as partes descritas a seguir são os passos fundamentais para a próxima etapa do trabalho.

A. Descrição e tratamento da base de dados

Os dados empregados neste projeto são provenientes da base aberta disponibilizada no Zenodo, denominada CODE15

[6]. Esta base é composta por arquivos no formato HDF5, que incluem o identificador do exame e os traçados correspondentes às 12 derivações do eletrocardiograma (ECG). Adicionalmente, há um arquivo no formato CSV contendo o identificador do paciente, o identificador do exame (idêntico ao dos arquivos HDF5), idade, sexo e o diagnóstico para seis condições cardíacas. Como parte do pré-processamento dos dados, foi aplicado um filtro para a correção da linha de base dos traçados do ECG [10]. O conjunto de dados abrange um total de 345 779 exames e 233 770 pacientes, sendo que cada um dos 12 traçados possui uma dimensão de 4096 pontos, com uma frequência de amostragem de 400 Hz. Atualmente, um processo de decimação foi aplicado no sinal, diminuindo a frequência de amostragem para 50Hz, totalizando 512 pontos da serie temporal.

B. Autoencoder

O *autoencoder* escolhido foi a arquitetura *Long Short-Term Memory* (LSTM), pelo fato que é um modelo próprio para trabalhar com séries temporais. Esse modelo tem o objetivo de extrair informações do ECG para o processamento nas próximas fases do projeto. Para a primeira parte, a LSTM é usada a implementação presente no pacote de python PyTorch, sendo ela descrita nas equações retiradas da documentação do pacote [11]:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}),\\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}),\\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}),\\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}),\\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t,\\
h_t &= o_t \odot \tanh(c_t),
\end{aligned}
$$

- h_t é o estado oculto no tempo t,
- c_t é o estado da célula no tempo t,
- x_t é a entrada no tempo t,
- h_{t-1} é o estado oculto da camada no tempo $t-1$ ou o estado oculto inicial no tempo 0,
- i_t, f_t, g_t, o_t são, respectivamente, os portões de entrada, esquecimento, célula e saída,
- σ é a função sigmóide,
- $\odot$ é o produto de Hadamard.

Como tanto o *encoder* quanto o *decoder* da arquitetura serão LSTM's, o h_t final do *encoder* será o input para o *decoder* , sendo o c_0 e x_0 do *decoder* inicializados como 0. A métrica de erro desse modelo é a norma L_1 do vetor de erro, que por sua vez é calculado subtraindo o tensor original da previsão.

Em conjunto com esse modelo, adicionamos camadas de convoluções com o intuito de melhorar o desempenho do modelo original, chegando à arquitetura final descrita na Figura 1. As camadas convolucionais foram construídas utilizando de três blocos principais, normalizações por batch, a função Gelu como ativação e a convolução (no caso do encoder) e a convolução transposta (no caso do decoder).

Para a implementação, utilizamos a biblioteca PyTorch, que oferece uma função robusta para convoluções 1D, descrita nas equações abaixo, conforme a documentação oficial [12]:

A operação de convolução 1D pode ser definida matematicamente da seguinte forma:

$$
\begin{aligned}
y(t) &= (x * w)(t) + b\\
&= \sum_{k=0}^{K-1} x(t+k)w(k) + b,
\end{aligned}
$$

onde:

- $y(t)$ é a saída no tempo t,
- $x(t)$ é a entrada no tempo t,
- $w(k)$ são os pesos do kernel de convolução,
- b é o viés,
- K é o tamanho do kernel.

A camada de convolução aplica o kernel w sobre a entrada x de forma deslizante ao longo de uma dimensão temporal, gerando uma nova representação y que captura padrões locais ao longo da sequência.

Para melhorar o desempenho e a estabilidade do treinamento, utilizamos normalizações por batch (*Batch Normalization*) e a função de ativação Gelu (*Gaussian Error Linear Unit*).

Normalização por Batch: A normalização por batch [13] é aplicada após a convolução para estabilizar e acelerar o treinamento. Ela normaliza a saída da convolução usando a média e o desvio padrão do mini-batch:

$$
\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{1}
$$

onde μ é a média do mini-batch, σ^2 é a variância, e ϵ é um valor pequeno para evitar divisão por zero. A normalização por batch ajuda a manter a ativação dos neurônios dentro de um intervalo estável, acelerando a convergência.

Função de Ativação Gelu: A função de ativação Gelu [14] é utilizada para introduzir não linearidades no modelo. Ela é definida como:

$$
\text{Gelu}(x) = x \cdot \Phi(x), \tag{2}
$$

onde $\Phi(x)$ é a função de distribuição cumulativa da normal padrão. A Gelu combina as vantagens das funções ReLU e Sigmoid, proporcionando um comportamento suave.

A combinação dessas técnicas garante um treinamento mais eficiente e robusto do modelo de convolução 1D.

Como hiper-parâmetros do experimento, foram utilizados tamanho latente da LSTM de 1024, cada LSTM tem 6 camadas, a taxa de aprendizado foi de 0.0003, treinado por 200 épocas. Além disso, as convoluções foram definidas para aumentar o número de canais, seguindo a sequência 12,64,64 e 128 e as convoluções transpostas foram definidas para diminuir o numero de canais seguindo a sequência 1024,512,256,128,12. O número de canais final do encoder e o número de canais inicial do decoder não são iguais pois

primeiro são aplicadas convoluções no encoder e depois a LSTM é aplicada, o que altera o número de canais. Logo, os canais subsequentes de ambas as arquiteturas foram escolhidos simplesmente como um decaimento/crescimento exponencial.

C. Mapa Causal

O mapa causal pode ser definido como um grafo acíclico e direcionado que modela as relações de causalidade em um conjunto de dados. Logo, uma aresta com inicio em A e final em B significa o atributo B está em função do atributo A, ou seja, A causa B. Ele também tem como parte sua definição o fato de ser um grafo direcionado e acíclico.

Para a definição do mapa causal, algumas estratégias foram testadas. A forma mais simples nesse caso seria fazer com que todos os atributos disponíveis (no nosso caso sexo, idade e diagnósticos) afetassem somente as ondas do sinal e não afetassem uns aos outros, como descrito na figura 2. Isso é equivalente a modelar um modelo generativo com os atributos atuando para condicionar a geração de amostras. Outra forma é utilizar o espaço latente do autoencoder em conjunto com os atributos já presentes para formar um grafo causal.

A segunda estratégia pode ser vista como mais interessante por revelar padrões mais profundos e significativos, porém não foi efeciente para o problema. A técnica utilizada foi o DAGMA [9], seguindo a implementação descrita no pacote python DAGMA , que é capaz de correlacionar padrões não lineares entre as variáveis fornecidas. Porém, ao treinar o DAGMA, o mapa causal retorna sem nenhuma aresta, independente das configurações e hiper-parâmetros testadas.

O algoritmo DAGMA consiste nas seguintes etapas:

1. **Definição do Modelo:** Um Autoencoder é definido para o conjunto de dados X, que deve estar no formato estruturado de uma matriz (N, M) com dados tabulares, onde N são as amostras e M são os atributos. O pacote Python utiliza uma rede MLP com camada de ativação sigmoide.

2. **Projeção no Espaço Latente:** Uma primeira camada densa é introduzida no modelo, projetando os atributos em um espaço latente de dimensões $(M \times D, M)$, onde M é o número de atributos e D é o tamanho do espaço latente.

3. **Treinamento do Modelo:** O modelo é treinado utilizando a função de perda de reconstrução, que é a média dos erros ao quadrado, somada com a normalização L_1 ou L_2 e, por fim, uma métrica de avaliação do mapa causal para garantir um grafo acíclico, a função $H_s(W)$.

Um dos entendimentos fundamentais para o algoritmo DAGMA é o conceito de matrizes M [15]. Matrizes M têm a forma $A = sI - B$, em que $B > 0$ e $s > \rho(B)$, em que $\rho(B)$ é o raio espectral da matriz B. Ao utilizar a função $H_s(W)$, os autores provam que o grafo induzido pelo modelo é um gráfico direcionado e acíclico e uma matriz M cujo raio espectral é inferior ao parâmetro s do algoritmo, que normalmente tem o valor de 1.

D. Função $H_s(W)$

A função $H_s(W)$ é essencial neste problema, pois o grafo causal deve ser acíclico. Portanto, treinamos o Autoencoder para que os pesos da primeira camada possam ser usados com esse objetivo. A função $H_s(W)$ foi introduzida com as características de ser sempre positiva e igual a 0 apenas quando o grafo induzido por W for acíclico.

A função $H_s(W)$ é definida como:

$$H_s(W) = -\log\det(sI - A) + d\log(s) \tag{3}$$

onde:

- s é um parâmetro escalar que controla o domínio das matrizes M, com valor padrão igual a 1.0.
- I é a matriz identidade de dimensão $d \times d$.
- A é uma matriz obtida a partir dos pesos da primeira camada densa ($fc1$), reorganizada em uma matriz de dimensão $M \times D \times M$, e somada ao quadrado ao longo da segunda dimensão, resultando em uma matriz de dimensão $M \times M$.
- d é o número de atributos.

O cálculo de A é realizado da seguinte forma:

$$A = \sum_{k=1}^{D} W_{ik}^2 \tag{4}$$

onde W_{ik} são os pesos da primeira camada densa reorganizados.

Assim, a função $H_s(W)$ é usada para garantir que o grafo causal induzido pelos pesos W seja acíclico, sendo sempre positiva e somente igual a 0 apenas quando o grafo induzido for acíclico.

E. Cálculo do Grafo Induzido

Para calcular o grafo induzido por W no algoritmo DAGMA, basta tomar C = $\sqrt{A^T}$

Com isso, o grafo causal será representado pela matriz C. A intuição deste processo é que estamos calculando a norma 2 do vetor de influência de cada atributo em relação a todos os outros atributos.

Com o algoritmo estabelecido, utilizou-se o encoder da primeira fase para extrair o espaço latente e criar a nova matriz de representação dos dados, que contém tanto o espaço latente quanto os atributos semânticos para a construção do grafo causal. Porém, não foi encontrada nenhuma relação causal em um nível tão granular de análise, onde tentamos estabelecer essas conexões entre cada um dos atributos latentes e os atributos semânticos (idade, sexo e diagnósticos).

Isso revela que, apesar de obviamente cada um dos atributos disponíveis afetar o sinal do ECG, não existe uma relação causal entre os atributos do estado latente e os atributos disponíveis em um nível individual. Ou seja, a relação causal nesse experimento só pode ser detectada com as técnicas empregadas a partir da análise do estado do ECG como um todo.

Logo, o mapa causal escolhido foi o método trivial, em que todos os atributos afetam o espaço latente do ECG, como exemplificado na figura 2.

IV. Resultados

Na figura 3 está o resultado da reconstrução do sinal do ECG nas suas 12 traçados, medido como a métrica L1 do sinal original menos o sinal reconstruído. A figura 3 mostra a distribuição do erro, com média de 566.3408 e desvio padrão de 561.5885. Apesar de desvio padrão e média elevadas, o fato que a medição está em relação em um sinal de 12 canais com 512 pontos para um total de 6144 pontos, ou seja, o erro de 566 esta dividido entre 6144, logo, temos um erro médio de 0.09 por ponto. Um exemplo da reconstrução do sinal esta disponível na figura 4.

V. Conclusão

O projeto tem objetivo de prever a evolução dos pacientes ao longo do tempo usando um extrator de informações e um modelo generativo para prever o próximo estado. O *Autoencoder* foi bem-sucedido na caracterização dos sinais em um espaço de baixa dimensionalidade, mas ainda é possível reduzir ainda mais a dimensionalidade e melhorar a reconstrução dos sinais. Na continuação do projeto, isso se tornará ainda mais importante devido ao fato de modelos generativos serem extremamente custosos de treinar, ou seja, qualquer redução de dimensionalidade será ainda mais valiosa. Além disso, como o mapa causal convergiu em uma simples injeção de atributos, o modelo generativo se torna mais simples de implementar, bastando projetar os atributos em um espaço de *embeddings* e injetar essas características no espaço latente. Por fim, a etapa final de prever a saúde do paciente com uma métrica baseada no espaço latente também se torna mais fácil com a menor dimensionalidade desse espaço, somada ao mapa causal simplificado.

Fig. 1. Arquitetura do Autoencoder

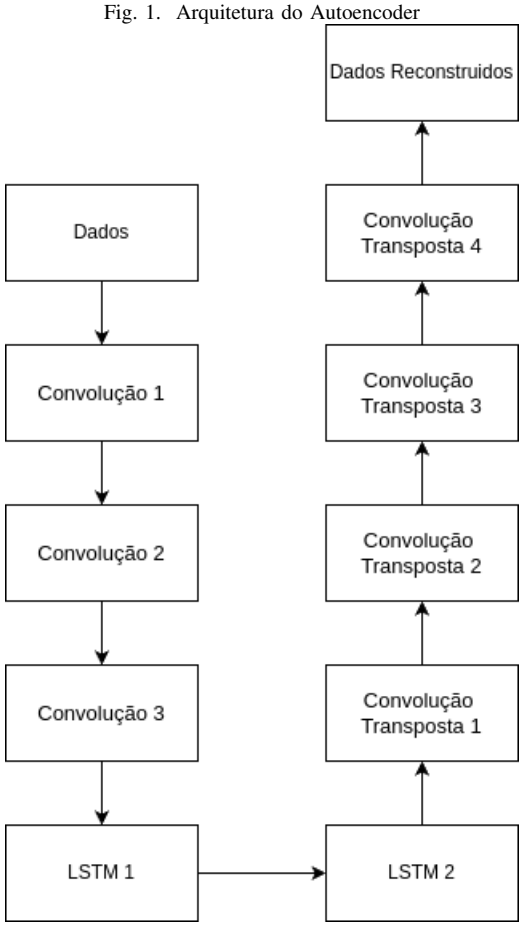


Fig. 2. Mapa Causal

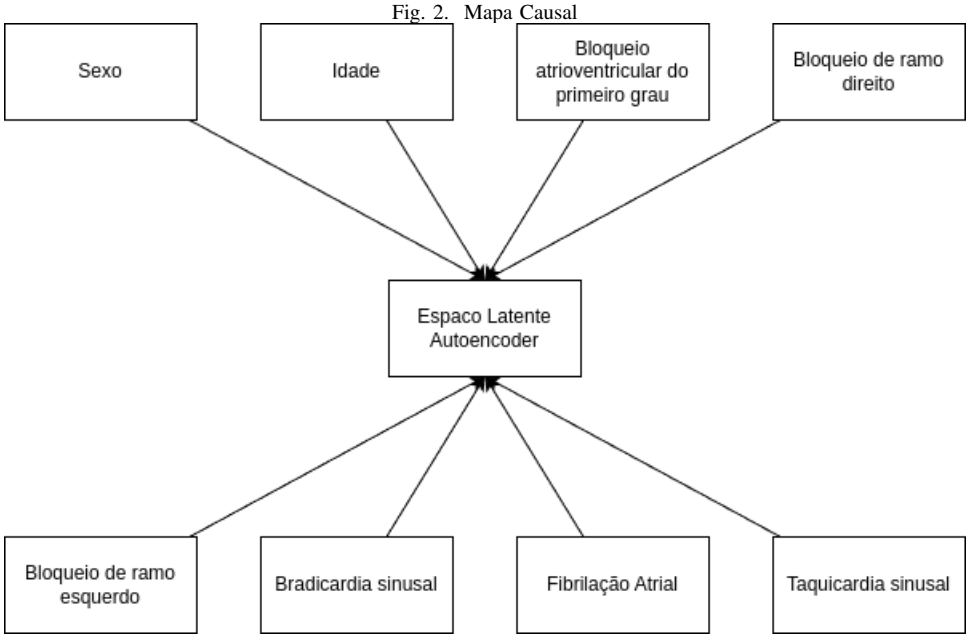


Fig. 3. Distribuição do erro

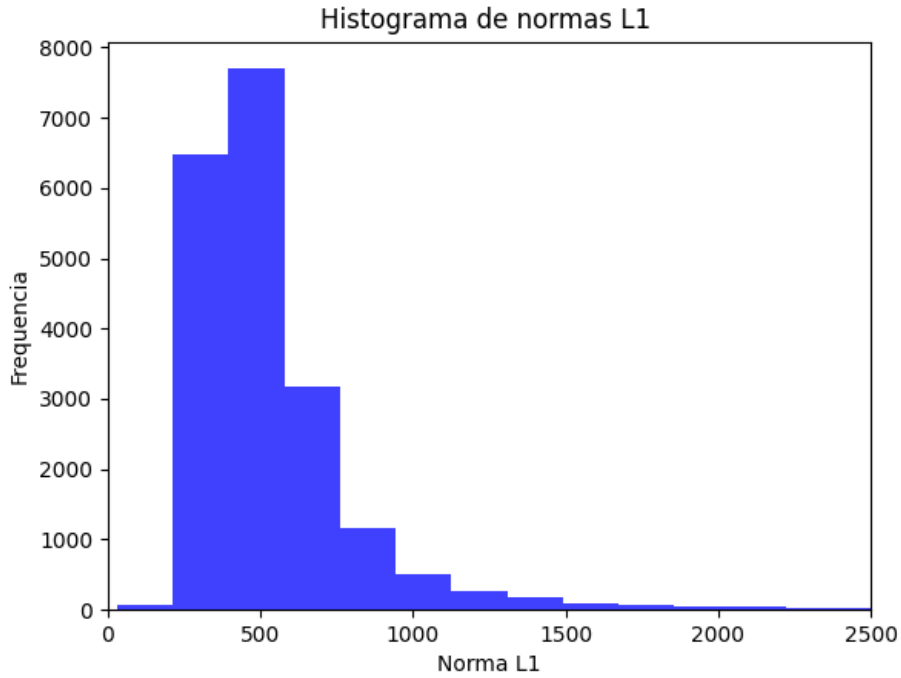
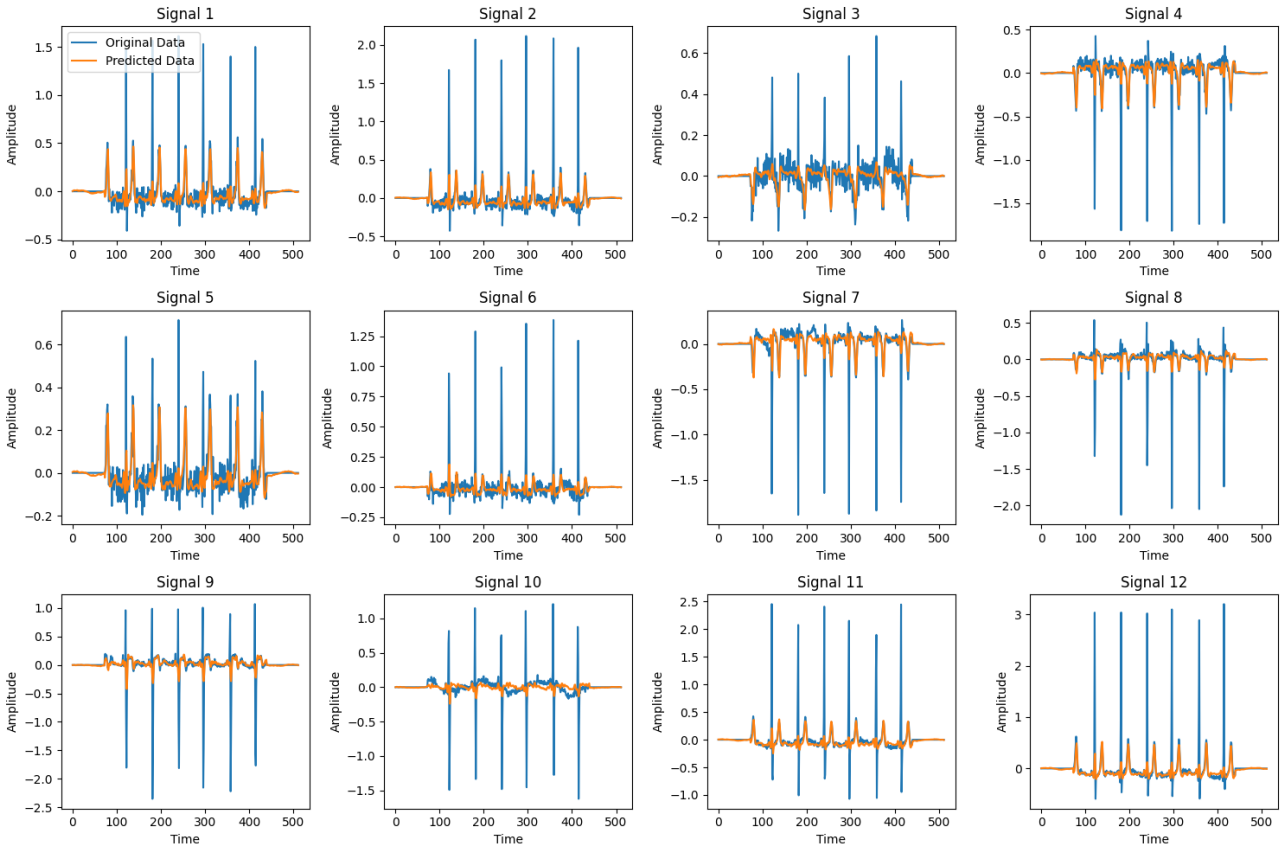


Fig. 4. Exemplo de reconstrução



REFERENCES

[1] S. G. Sepanlou, S. Safiri, C. Bisignano, K. S. Ikuta, S. Merat, M. Saberifiroozi, H. Poustchi, D. Tsoi, D. V. Colombara, A. Abdoli *et al.*, "The global, regional, and national burden of cirrhosis by cause in 195 countries and territories, 1990–2017: a systematic analysis for the global burden of disease study 2017," *The Lancet gastroenterology & hepatology*, vol. 5, no. 3, pp. 245–266, 2020.

[2] J. Schläpfer and H. J. Wellens, "Computer-interpreted electrocardiograms: benefits and limitations," *Journal of the American College of Cardiology*, vol. 70, no. 9, pp. 1183–1192, 2017.

[3] N. Pawlowski, D. Coelho de Castro, and B. Glocker, "Deep structural causal models for tractable counterfactual inference," *Advances in neural information processing systems*, vol. 33, pp. 857–869, 2020.

[4] J. Pearl and D. Mackenzie, *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018.

[5] Y. Ming, H. Yin, and Y. Li, "On the impact of spurious correlation for out-of-distribution detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, no. 9, 2022, pp. 10 051–10 059.

[6] A. H. Ribeiro, G. M. M. Paixao, E. M. Lima, M. Horta Ribeiro, M. M. Pinto Filho, P. R. Gomes, D. M. Oliveira, W. Meira Jr, T. B. Schon, and A. L. P. Ribeiro, "CODE-15%: a large scale annotated dataset of 12-lead ECGs," 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4916206

[7] X. Xiong and L. Luo, "Inpatient flow distribution patterns at shanghai hospitals," *International Journal of Environmental Research and Public Health*, vol. 17, no. 7, p. 2183, 2020.

[8] F. Draxler, P. Sorrenson, L. Zimmermann, A. Rousselot, and U. Köthe, "Free-form flows: Make any architecture a normalizing flow," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2024, pp. 2197–2205.

[9] K. Bello, B. Aragam, and P. Ravikumar, "Dagma: Learning dags via m-matrices and a log-determinant acyclicity characterization," *Advances in Neural Information Processing Systems*, vol. 35, pp. 8226–8239, 2022.

[10] P. Gupta, K. K. Sharma, and S. D. Joshi, "Baseline wander removal of electrocardiogram signals using multivariate empirical mode decomposition," *Healthc Technol Lett*, vol. 2, no. 6, pp. 164–166, 2015.

[11] PyTorch, "torch.nn.lstm — pytorch documentation," https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html, 2024, accessed: 2024-04-03.

[12] P. Documentation, "torch.nn.conv1d," *PyTorch Documentation*, 2023. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html

[13] ——, "torch.nn.batchnorm1d," *PyTorch Documentation*, 2023. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm1d.html

[14] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016. [Online]. Available: https://arxiv.org/abs/1606.08415

[15] A. Ostrowski, "Über die determinanten mit überwiegender hauptdiagonale," *Commentarii Mathematici Helvetici*, vol. 10, no. 1, pp. 69–96, 1937.