UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

PEDRO FRANCISCO GODOY BERNARDINELLI

Atenção exponencial: um novo método de aprendizado profundo para previsão em séries temporais usando expoentes de Hurst

Campinas
21/11/2024

PEDRO FRANCISCO GODOY BERNARDINELLI

Atenção exponencial: um novo método de aprendizado profundo para previsão em séries temporais usando expoentes de Hurst*

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado, sob a orientação do(a) Prof. João Batista Florindo.

*Este trabalho foi financiado pela PIBIC/CNPq.

Resumo

O projeto analisou o desempenho de redes neurais profundas no problema de previsão de séries temporais. Mais precisamente, foram empregados um modelo denso totalmente conectado e um modelo do estado-da-arte na área (N-BEATS). Para tal, foram utilizadas duas séries temporais com características contrastantes e, para cada uma delas, foram realizados três experimentos com configurações diferentes para os modelos. Os resultados obtidos foram comparados para obter uma variedade de conclusões a respeito das séries e dos modelos: o desempenho dos modelos em uma série temporal mais bem comportada é superior à performance em uma série volátil; o impacto da escolha dos hiperparâmetros nos resultados é significativo; a variação da série temporal e da configuração causa uma grande variabilidade nos resultados, com a alternância de qual modelo teve melhor desempenho em cada caso; o aumento do horizonte de previsão dificulta o problema, com resultados piores nessas circunstâncias e o impacto do aumento da quantidade de dados fornecidos aos modelos varia conforme a série temporal, não sendo possível apresentar uma resposta definitiva sobre isso.

Abstract

The project analyzed the performance of deep neural networks on the time series forecasting problem. More precisely, a fully connected dense model and a state-of-the-art model in the area, N-BEATS, were employed. For such, two time series with opposing characteristics were used and, for each one, three experiments with different settings were made for each model. The results were compared to reach a variety of conclusions regarding the time series and the models: the performance of the models is superior on a better behaved time series if compared with volatile series; the choice of hyperparameters causes a significant impact on the results; the variation of the time series and of the experimental configuration causes a great variability in the results, changing which model performs better; the increase of the forecasting horizon complicates the problem, with worse results under these circumstances and the impact of increasing the amount of data provided to the models varies according to the time series, not allowing a definitive answer to this question.

Conteúdo

1 Introdução **6**

2 Materiais e modelos **7**

2.1 Problema de previsão 7

2.2 Problema de classificação 8

2.3 Expoente generalizado de Hurst 10

2.4 Mecanismo de auto-atenção 12

2.5 Redes Neurais Totalmente Conectadas 13

2.6 N-BEATS 15

2.7 Modelo proposto 16

3 Resultados **18**

3.1 Preço do *Bitcoin* 18

3.2 Classificação dos eletrocardiogramas 19

4 Conclusão **20**

1 Introdução

O presente projeto tem como objetivo introduzir um novo modelo de aprendizado profundo para realizar tarefas relacionadas a séries temporais. O desempenho de tal modelo foi explorado tanto em problemas de previsão quanto de classificação.

Séries temporais estão presentes em quase todos os ramos do mundo contemporâneo: na bolsa de valores, na previsão do tempo, nos índices de produção de energia elétrica, em exames médicos (como o eletrocardiograma), entre tantos outros. Sendo assim, é natural buscar formas de melhorar os resultados obtidos atualmente para a previsão ou para a classificação de tais dados. Tendo em vista os avanços assustadoramente vertiginosos que têm ocorrido nos últimos anos na inteligência artificial, sobretudo com o aprendizado de máquinas, é fácil imaginar o interesse em juntar esses dois temas: séries temporais e *machine learning*. Assim, há inúmeras pesquisas para a implementação de modelos (sejam eles criados ou simplesmente adaptados de outros problemas) para tratar de séries temporais, como [7, 10, 16] entre muitos outros.

Os problemas de previsão são encontrados frequentemente em questões financeiras, como no preço de ações ao longo do tempo. Sendo assim, a previsão de séries temporais tem um interesse muito claro neste contexto: prever os próximos preços de certa ação. Entretanto, não há apenas considerações financeiras para a previsão de séries temporais: previsões acuradas da geração de energia elétrica nos próximos meses ou dos índices de precipitação futuros permitem a criação de planos de contigência para eventuais faltas de água ou de energia.

Já a classificação de séries temporais possui uma aplicação direta em áreas como a medicina: por exemplo, na detecção automática de anomalias cardíacas a partir de eletrocardiogramas, o que permitiria uma maior acurácia e uma maior agilidade na detecção de tais problemas, agilizando no tratamento e prevenindo o agravamento das condições.

Destarte, é evidente que há um grande interesse (e potencial) para o *machine learning* auxiliar nesses vários campos. Portanto, este projeto estuda como a junção entre um modelo do estado-da-arte (N-BEATS, [10]), um mecanismo de auto-atenção ([14]) e um medidor estatístico da dependência de longo-prazo de uma série temporal (expoente

generalizado de Hurst [6, 8]) podem melhorar o desempenho do modelo básico, que usaria apenas o N-BEATS padrão.

2 Materiais e modelos

Nesta seção, serão apresentados as diferenças entre os dois problemas e os componentes do modelo proposto.

2.1 Problema de previsão

Há diferenças entre notórias entre os dois problemas que serão resolvidos aqui. Assim, é importante introduzí-los antes de mais nada.

O problema de previsão consiste em prever os próximos valores de uma série temporal a partir de uma janela anterior de tal. Assim, em um modelo de previsão é passado como dado de entrada uma janela da série temporal de tamanho pré-determinado e o modelo deve retornar previsões para os valores de um horizonte pré-definido da série temporal.

No caso analisado neste projeto, foi utilizada a série dos preços diários do Bitcoin disponíveis em [3], o qual também possui o código do N-BEATS utilizado como base paa a implementação do presente modelo. Assim, como há apenas uma série temporal, esta mesma série será utilizada para treino e teste do modelo. Como, obviamente, os mesmos dados não podem er utilizados para treino e teste, optou-se por dividir a série temporal em duas partes: os primeiros 80% dos dados foram utilizados para treino e o restante para teste. Não obstante, em cada um dos dois conjuntos, os dados foram divididos em "janelas deslizantes"percorrendo todo o conjunto de dados, onde a janela, de tamanho fixo, era a entrada e os próximos valores da série temporal eram a saída esperada do modelo.

Essa mesma série temporal foi utilizada no projeto do ano passado [1], onde há uma análise detalhada das características da série temporal. Aqui não há necessidade de tantos detalhes, mas uma visualização gráfica da série e algumas informações básicas dela são importantes para conseguir analisar os resultados de forma mais completa:

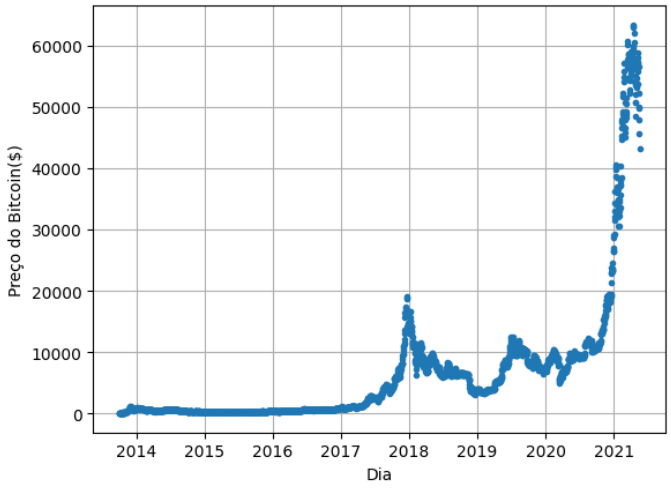

Figura 1: Série temporal para o problema de previsão.

Média	6529.845
Desvio padrão	10887.847
Máximo	63346.789
Mínimo	108.585

Tabela 1: Informações básicas sobre a série temporal do preço dos Bitcoins.

É fácil ver que trata-se de uma série temporal com um comportamento bastante errático e com padrões diferentes no começo e no final, o que danifica severamente os resultados do modelo, pois o conjunto de treinamento usa o começo, mas o teste usa o final da série temporal, ou seja, os padrões identificados no treinamento são alterados no teste. Entretanto, problemas do cotidiano ocorrem desta forma, já que os preços de ações decorrem de inúmeros outros fatores, o que afeta o comportamento dele com o tempo.

2.2 Problema de classificação

O problema de classificação é fundamentalmente diferente. Nele, dada uma série temporal inteira (aqui, um eletrocardiograma), o modelo deve classificar a série temporal entre diversas classes. No caso tratado aqui, as classes (no caso, anomalias cardíacas) não são mutualmente exclusivas, ou seja, uma mesma série temporal pode estar em qualquer número de classes (0, 1 ou múltiplas).

O conjunto de dados utilizado para o treinamento e para o teste do modelo é o conjunto de dados utilizado no Desafio de Computação na Cardiologia de 2021 (Com-

puting in Cardiology Challenge 2021 [4]). Entretanto, não foram utilizados a totalidade dos exemplos disponíveis: apenas aqueles cujas séries temporais possuem tamanho 60000, o que, corresponde a um eletrocardiograma com 12 leads cada uma com 10 segundos de duração e 500 Hz de frequência. Isso corresponde a maioria dos dados disponíveis, totalizando 74.374 exemplos correspondentes as seguintes bases de dados: Chapman-Shaoxing (Chapman University e Shaoxing People's Hospital); Ningbo (Ningbo First Hospital) ([18, 17]; Physikalisch-Technische Bundesanstalt (PTB) (no caso foi usado somente a PTB-XL) ([15]) e uma base de dados do estado da Georgia, Estados Unidos. As classes não são balanceadas, havendo certas anomalias cardíacas raras com apenas 1 exemplo no conjunto de dados. Isso quer dizer que há classes em que não se espera que o modelo seja capaz de prever corretamente.

Todos essas bases de dados foram juntadas e misturadas, sendo, por fim, divididas em 5 *folds* diferentes. Sendo que para cada treinamento e teste do modelo, um *fold* diferente era separado para teste e todos os outros eram os exemplos de treinamento. Desse modo, todos os exemplos foram usados tanto para teste quanto para treino.

Como exemplo, segue um dos exemplos de eletrocardiograma usado:

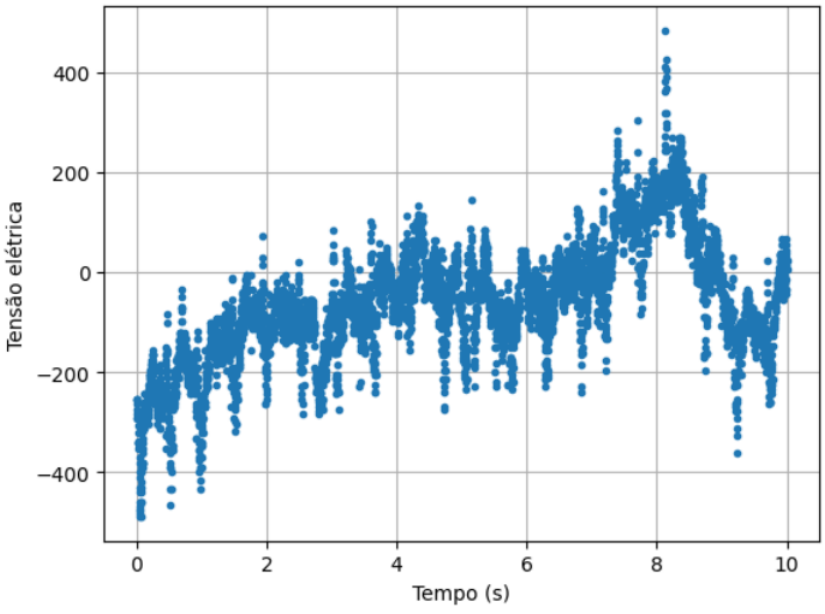

Figura 2: Eletrocardiograma.

2.3 Expoente generalizado de Hurst

O expoente de Hurst (H) é uma medida da dependência de longo prazo de uma série temporal. A dependência de longo prazo relaciona-se com a taxa de decaimento da autocorrelação de dois pontos da série temporal com intervalos cada vez maiores: se esse decaimento é mais lento do que uma exponencial, então diz-se que a série temporal possui dependência de longo-prazo. Intuitivamente, isso significa que valores futuros são afetados por valores passados da série temporal. O expoente é uma homenagem ao hidrologista Harold Edwin Hurst (1880-1978), o qual liderou a pesquisa para encontrar o tamanho ótimo da represa no rio Nilo e concluiu que os níveis de água do rio possuiamdependência de longo-prazo [6].

O expoente de Hurst pode ser generalizado para uma forma mais geral: o expoente generalizado de Hurst (GHE, de *Generalized Hurst Exponent*) é uma medida particular derivada da teoria de sistemas dinâmicos não lineares e que serve como um indicativo da dependência estatística de longo-prazo presente em uma série temporal $X(t)$, definida em uma janela de tempo Δt, sendo $t = (1, 2, ..., \Delta t)$. A medida é baseada nos q-ésimos momentos (K_q) da distribuição dos incrementos dos dados da série temporal [9]:

$$K_q(\tau) = \frac{\langle |X(t+\tau) - X(t)|^q \rangle}{\langle |X(t)|^q \rangle} \tag{1}$$

Na Equação (1), τ é um parâmetro que varia de 1 até τ_{max} e $\langle . \rangle$ representa a média da medida no interior dos parênteses angulados na janela de tempo sendo considerada.

Assim, o valor do expoente generalizado de Hurst ($H(q)$) é definido de acordo com a lei de potência de $K_q(\tau)$:

$$K_q(\tau) \propto \tau^{qH(q)} \tag{2}$$

Logo, observe-se que há duas situações possíveis para $H(q)$:

- (I): $H(q)$ é constante para todo q, ou seja, $H(q) = H$, onde H é o próprio expoente de Hurst ou índice de auto-afinidade - processos com essa característica são chamados de unifractais;
- (II): $H(q)$ não é constante e apresenta valores diferentes para os diferentes momentos

- esses processos são chamados de multifractais.

Em geral, as séries temporais que serão analisadas neste projeto entram no segundo caso.

$H(q)$ assume valores entre 0 e 1 e, de modo geral, para $H(q) > 0,5$, temos que as flutuações da série temporal relacionadas à ordem q são persistentes (ou seja, se a série está crescendo, ela tem a tendência de continuar crescendo e vice-versa). Já para $H(q) < 0,5$, as flutuações são anti-persistentes (há a tendência de parar de crescer e começar a decrescer). Finalmente, para $H(q) = 0.5$, não é possível afirmar se a tendência é continuar crescendo ou inverter o sentido da variação.

O cálculo do GHE é feito utilizando uma linearização da Equação (2). Para um dado q é feito um gráfico de $\log(K_q(\tau))$ por $q\log(\tau)$, ou seja, uma curva log-log. Esse gráfico deve ficar similar a uma reta. O coeficiente generalizado de Hurst é o coeficiente angular da reta que melhor se ajusta ao gráfico. Tal valor geralmente é calculado por meio de mínimos quadrados.

Assim, o código base utilizado para esse cálculo encontra-se no seguinte GitHub [13], o qual realiza o cálculo do GHE por meio de mínimos quadrados dada uma série temporal (ou uma janela desta) e um valor de q, que indica qual é o momento da distribuição sobre o qual o expoente deve ser calculado.

Nesse código, é calculado um GHE para um dado valor de q e uma certa janela de uma série temporal (para o cálculo ter algum sentido, é necessária uma janela de tamanho maior que 150). Foram calculados q de 1 até 8 para cada lead de cada exemplo, ou seja, para cada exemplo haviam 12 valores expoentes para cada q, os quais foram armazenados em uma matriz tridimensional.

Entretanto, o código possuía um problema grande: ele realizava o código para apenas um expoente de uma janela de um exemplo de cada vez, oq ue causava uma grande demora. Para amenizar tal efeito, foram feitas alterações no código para realizar o código de forma paralelizada, situação mais rápida na linguagem de programação escolhida: Python. As alterações não se tratam de mudanças na forma de calcular, as etapas ainda são as mesmas do código base, a única alteração foi para permitir que esse cálculo fosse feito em vários exemplos ao mesmo tempo, mas ainda com uma janela por exemplo e um mesmo q para todos. Isso acelerou de forma perceptível o código, mas não o suficiente

para impedir que se tornasse um gargalo para o modelo. De modo geral, para cada 1000 exemplos é necessário esperar 1 hora para obter os expoentes.

Embora isso seja um problema, esses expoentes são calculados uma única vez para cada exemplo, assim, esse é um obstáculo apenas no começo, após os cálculos dos expoentes é só salvá-los em um arquivo, não sendo necessário calculá-los todas as vezes.

Essa matriz será utilizada como entrada para o mecanismo de auto-atenção.

2.4 Mecanismo de auto-atenção

Mecanismos de atenção começaram a ganhar grande popularidade a partir do estudo em [14], o qual introduziu a arquitetura *transformer* e a ideia de *self-attention*. Embora essa arquitetura em específico não seja utilizada neste projeto, o mecanismo de atenção proposto é fundamental na metodologia proposta.

Intuitivamente, o mecanismo de auto-atenção é responsável por determinar quais partes dos dados o modelo deve prestar mais atenção para melhorar a predição do horizonte desejado. Assim, o mecanismo de atenção é responsável por definir pesos para cada parte da série temporal. Como o nome sugere, o mecanismo de auto-atenção utiliza a própria série temporal para descobrir em qual parte da mesma se deve prestar mais atenção (ou seja, atribuir um peso maior).

Apresentada essa noção mais intuitiva do modelo de auto-atenção, será detalhado a seguir como o cálculo funciona de modo mais técnico.

Suponha que temos os dados de entrada armazenados em uma matriz A. É a partir dessa matriz A que serão aprendidos os pesos do mecanismo de auto-atenção. O primeiro cálculo que deve ser feito é o produto matricial de A com três matrizes de parâmetros (os quais devem ser aprendidos pelo modelo) W^q, W^k e W^v. O resultado desses produtos são, respectivamente, as matrizes: *Query* (Q), *Key* (K) e *Value* (V).

A próxima etapa é fazer o produto interno entre Q e K, ou seja, realizar o produto matricial QK^T. A intuição por trás desse cálculo é que o produto interno está medindo a similaridade entre os valores de cada linha de Q e de cada linha de K (que são as colunas de K transposto). Uma função *softmax* é então aplicada sobre a matriz resultante dividida pela raiz da dimensão de Q (essa divisão é apenas para normalizar as entradas). Essa função transforma todas as entradas da matriz em números entre 0 e 1,

sendo que a soma de cada linha resulta em 1.

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{3}$$

Na Equação ((3)), z é um vetor qualquer, i é uma componente do vetor e N é o tamanho do vetor.

Finalmente, após a aplicação da função *softmax*, há apenas mais um cálculo necessário para encontrar a saída dessa *head* do mecanismo de atenção (o que é uma *head* será explicado logo após esse cálculo). Essa última operação é simplesmente o produto entre a matriz após a *softmax* e a matriz V. Nesse caso, está sendo realizada uma soma ponderada dos valores de V, em que os pesos são simplesmente cada coluna da *softmax*.

Destarte, juntando todas as operações descritas nos parágrafos anteriores, temos que o cálculo total se resume a:

$$\text{Attention} = \text{softmax}(\frac{(AW_q)(AW_k)^T}{\sqrt{d_K}})(AW_v). \tag{4}$$

O resultado de (4) é a saída de uma *head*. O modelo que será estudado neste projeto é um *Multi-Head Self-Attention*, assim, é necessário repetir o cálculo exibido anteriormente várias vezes, com matrizes de parâmetros W^q, W^k e W^v diferentes para cada vez. Cada saída recebe o nome de *head* e, no final, após o cálculo de todas as *heads*, elas são concatenadas e temos a saída final do mecanismo de atenção.

No modelo proposto nesse artigo, o mecanismo de auto-atenção será utilizado usando a matriz proveniente do GHE como entrada.

2.5 Redes Neurais Totalmente Conectadas

Antes de introduzir a arquitetura do estado-da-arte que será utilizada para fazer a classificação dos eletrocardiogramas e a previsão de séries temporais, é importante apresentar o modelo mais simples de redes neurais: as redes neurais totalmente conectadas [2]. Uma rede neural genérica pode ser vista na Figura 3, cujos elementos serão detalhados nos próximos parágrafos:

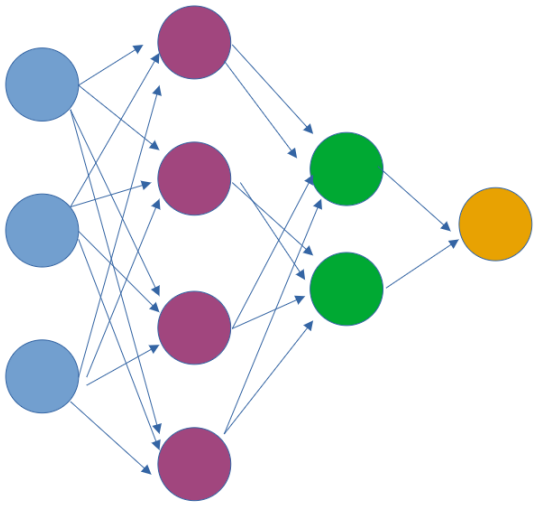

Figura 3: Representação de uma rede neural totalmente conectada pequena

Cada cor representa uma camada diferente da rede neural. Uma camada recebe os valores da camada anterior, realiza cálculos com eles e propaga os resultados para a próxima camada. A camada de cor azul é a camada de entrada dos dados, ela tipicamente não é considerada para a profundidade da rede (a quantidade de camadas da rede), pois, de modo geral, ela apenas recebe os dados de entrada, sem realizar nenhum cálculo com eles.

Já a camada roxa é a primeira camada propriamente dita, a camada verde é a segunda e a laranja é a terceira e última. O resultado das operações na última camada é a saída do modelo. Portanto, essa arquitetura representada na imagem é uma rede neural de 3 camadas.

Cada círculo colorido é uma unidade de ativação, chamados também de "neurônios". Em cada neurônio ocorrem 2 operações diferentes: a primeira é a combinação linear dos resultados dos neurônios da camada anterior ponderados por pesos. Esses pesos são os parâmetros de uma rede neural totalmente conectada e são aprendidos durante o treinamento. O resultado da combinação linear é utilizado como argumento de uma função não linear. Tal função não-linear é chamada de função de ativação, e há várias opções: tangente hiperbólica, sigmoide, entre outras. Em redes mais profundas, a função de ativação que é mais utilizada é a ReLu (*Rectified Linear Unit*):

$$\text{ReLU}(z) = \max\{0, z\} \tag{5}$$

Observe que tal função é trivialmente calculada e seu principal papel é introduzir uma não-linearidade à rede neural, o que permite que ela aprenda coisas mais interessantes. Além disso, essas operações são realizadas para todos os neurônios de todas as camadas.

Por fim, o resultado da função de ativação é a saída desse neurônio, a qual será propagada para os neurônios de próxima camada. Note, que o nome de redes neurais totalmente conectadas decorre do fato de que nessas arquiteturas todos os neurônios de uma camada estão conectados com todos os neurônios da camada subsequente.

Esse processo de, a partir dos dados de entrada, calcular a saída de cada neurônio e, por fim, a saída da rede é chamado de *forward propagation*. Para aprender os parâmetros, é utilizado o caminho inverso, partindo da saída da rede até a primeira camada e tal algoritmo é chamado de *backpropagation*, que por sua vez pode usar o clássico algoritmo de otimização do gradiente descendente, assim como versões mais elaboradas deste como Adam, RMSProp, etc.

As redes neurais totalmente conectadas, embora tenham uma definição aparentemente simples, são extremamente úteis e constituem a base de inúmeros modelos mais avançados e complexos do estado-da-arte, entre eles, o N-BEATS que será usado neste projeto.

2.6 N-BEATS

O principal modelo que será utilizado aqui para realizar a classificação e a previsão é o N-BEATS (*Neural Basis Expansion Analysis for Interpretable Time Series* ou Análise de Expansão de Base Neural para Séries Temporais Interpretáveis), aqui devidamente adaptado para o mecanismo de auto-atenção implementado. O modelo básico do N-BEATS ([10]) é basicamente um conjunto de L blocos residuais. Cada bloco l é um mapeamento não-linear com camadas totalmente conectadas que devolve duas saídas distintas: uma saída que fará parte da predição final (y_l) e uma saída que será usada para compor a entrada do próximo bloco ($\hat{x}_l$). No primeiro bloco, a entrada é a própria série temporal, cujos próximos valores serão previstos pelo modelo. Ao final de todos os blocos, as saídas (y_l) são somadas e formam a predição para o próximo valor da série temporal (ou para os próximos valores, dependendo do horizonte do problema), ou seja, a saída

predita pelo modelo $\hat{y}$ é:

$$\hat{y} = \sum_{l=1}^{l=L} y_l. \tag{6}$$

Além disso, as entradas do bloco l para $1 < l \leq L$ são:

$$x_l = x_{l-1} - \hat{x}_{l-1}. \tag{7}$$

Traduzindo em palavras, a entrada de um bloco é simplesmente a diferença entre a entrada e uma das saídas do bloco anterior. Esse tipo de bloco é chamado de bloco residual ([5]).

Um diagrama para auxiliar na compreensão do modelo é apresentado na Figura 4.

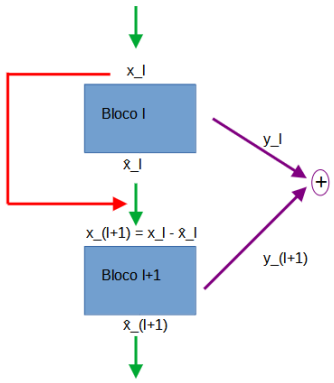

Figura 4: Diagrama de como os dados passam pelo N-BEATS.

Embora originalmente o N-BEATS tenha sido proposto para um problema de previsão de séries temporais, o uso dele em problemas de classificação não é novidade, em especial [11, 12] apresentam resultados utilizando a mesma base de dados de eletrocardiogramas usada no presente projeto.

2.7 Modelo proposto

Todos os elementos apresentados acima são as diversas partes que compõem o modelo proposto. Contudo, observe que não houve nenhuma diferenciação entre os modelos de classificação e de regressão. Isso ocorre porque os modelos são muito parecidos, sendo que as duas principais diferenças encontram-se em detalhes apresentados nos

parágrafos subsequentes.

Para o modelo de previsão, o GHE é calculado para toda a série temporal em janelas de tamanho pré-fixado e até um expoente também pré-determinado (no caso, 8). Essa matriz com os expoentes é passada como entrada para o mecanismo de auto-atenção, essa "atenção exponencial" retorna uma matriz com os valores de atenção. Essa matriz é então passada por uma pequena rede neural totalmente conectada de duas camadas para ajustar suas dimensões para o N-BEATS.

Após essa pequena rede neural, a saída de tal é concatenada com uma janela dos dados. Esse novo elemento concatenado é então utilizado como entrada para o N-BEATS. É importante ressaltar que a saída da rede neural é concantenada com a entrada de cada bloco do N-BEATS, então tal informação percorre o modelo inteiro. Finalmente, a saída predita é simplesmente a soma das saídas de cada bloco.

Todo este modelo pode ser visualizado na Figura 5:

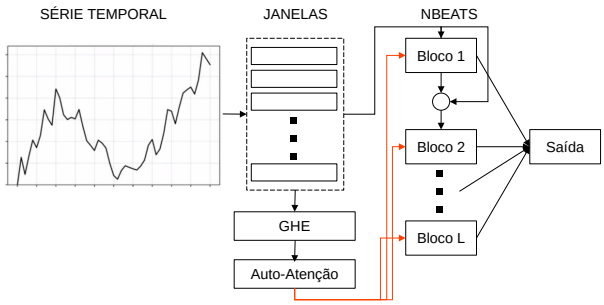

Figura 5: Modelo proposto: as informações da série temporal são divididas em janelas onde o GHE é calculado para vários expoentes. Tais valores são passados para o mecanismo de auto-atenção. Por fim, a série temporal e a saída do mecanismo de auto-atenção são concatenados e usados como entrada para o N-BEATS.

Já no modelo de classificação, antes de mais nada, é feito um *downsample* de 25× sobre todos os exemplos. Após isso, o GHE é calculado para cada *lead* de cada eletrocardiograma para cada valor possível do expoente. Note, que diferentemente do caso de regressão, onde há apenas uma matriz de GHE já que há uma única série temporal, aqui cada eletrocardiograma tem sua própria matriz de GHEs. Essas matrizes são agrupadas então em uma matriz tridimensional.

O restante do procedimento é igual ao modelo de previsão até a saída da rede. Aqui, a saída da rede tem mais uma etapa após a soma da saída de cada bloco: é aplicada

uma função sigmoide para colocar os valores entre 0 e 1 para cada classe. Por fim, se o valor predito for maior que um certo *threshold*, então o eletrocardiograma em questão pertence a tal classe e, do contrário, não pertence a tal.

A Figura 5 também ilustra bem o modelo de classificação. No caso, as janelas para o GHE seriam os *leads* de cada eletrocardiograma que são concantenadas para o modelo N-BEATS.

3 Resultados

3.1 Preço do *Bitcoin*

Primeiramente, será testado o modelo para a previsão do preço do *Bitcoin*. Para tal, será apresentado o resultado para o modelo N-BEATS básico e o modelo proposto lado a lado para essa série temporal, possibilitando uma comparação e uma análise dos ganhos possíveis com a inovação proposta. Cada modelo foi treinado 5 vezes e foi feita uma média aritmética simples dos resultados para obter os valores finais, isso foi feito para diminuir a influência de aleatoriedades da inicialização dos pesos de treinamento. Foi utilizada uma janela com as 7 últimas entradas da série temporal e um horizonte de previsão unitário. Abaixo estão os resultados no conjunto de teste:

Métrica	N-BEATS	Modelo proposto
MAE	578.968	625.898
RMSE	1085.529	1154.5
MAPE	2.649	2.830
MASE	1.017	1.099

Tabela 2: Resultados dos experimentos para o preço do Bitcoin.

As métricas apresentadas são o MAE (erro absoluto médio, é a média dos erros em módulo), o RMSE (raiz quadrada da soma dos quadrados dos erros), o MAPE (erro porcentual absoluto médio) e MASE (erro médio absoluto escalado). O MASE em especial é bastante interessante, já que trata-se de uma métrica que compara o desempenho do modelo atual com um preditor muito simples, que consiste em prever para o próximo valor simplesmente o valor anterior. Assim, qualquer valor de MASE maior do que 1 implica que o modelo é pior do que esse preditor simples. Bem, como pode ser visto na

tabela acima, ambos os modelos tiveram resultados maiores do que 1. Contudo, isso é algo comum em mercado financeiro, já que os problemas costumam ser complicados. Portanto, esse desempenho decepcionante não é algo surpreendente ou não usual. Explicações mais detalhadas sobre as métricas de avliação podem ser vistas no projeto anterior [1].

Assim, analisando os resultados da Tabela 2, fica evidente que ambos os modelos sofreram muito para aprender corretamente os padrões da série. Isso se explica pela complexidade e pela alta variabilidade dos valores. Entretanto, claramente, o modelo proposto não obteve o desempenho esperado, já que a introdução do GHE e do mecanismo de auto-atenção não causou uma melhoria no modelo básico, pelo contrário, os resultados foram piores em todas as métricas. No entanto, há uma explicação razoável para tal: o GHE, da forma que foi utilizado e calculado, apresenta informações da série temporal inteira e não apenas da janela que está sendo analisada pelo modelo. Assim, é possível que as informações de fora da janela estejam confundindo o modelo. Espera-se que isso não ocorra no problema de classificação, já que em tal situação a série temporal inteira é analisada.

3.2 Classificação dos eletrocardiogramas

Como explicado anteriormente, para o problema de classificação, o conjunto total de dados foi dividido em 5 conjuntos menores (chamados *folds*). Cada conjunto foi utilizado uma vez para teste e quatro vezes para treinamento, resultando em 5 modelos treinados e avaliados no conjunto de teste correspondente. No final, foi feita uma média aritmética entre os 5 modelos e o resultado final é o apresentado a seguir:

Métrica	N-BEATS	Modelo proposto
Precision	0.4067	0.4418
Recall	0.2759	0.2806
Accuracy	0.1991	0.2299
F1-Score	0.3286	0.3430

Tabela 3: Resultados dos experimentos para a classificação dos eletrocardiogramas.

Novamente, eles foram dispostos lado a lado para facilitar uma comparação direta entre os dois modelos. Antes de mais nada, uma breve apresentação das métricas de avaliação: *Precision* é uma medida da confiabilidade das classes preditas como positivas,

ou seja, ela retorna a proporção de quantas predições positivas realmente o são; *Recall* é uma medida da capacidade de identificar os casos positivos, é a proporção de quantos dos casos realmente positivos o classificador identificou corretamente; *Accuracy*, a acurácia, é a porcentagem de quantos exemplos foram classificados corretamente em todas as 112 classes; *F1-Score*, uma fórmula para tentar reduzir a avaliação de um modelo em um único número:

$$\text{F1-Score} = 2\frac{PR}{P+R}, \tag{8}$$

em que P é o *Precision* e R é o *Recall*.

Conhecidas as métricas apresentadas, é fácil ver, a partir da Tabela 3 que os resultados do modelo proposto são notavelmente superiores ao modelo N-BEATS tradicional. Em todas as métricas, a adição da "atenção exponencial" causou uma melhoria na performance do modelo. Assim, claramente, há um ganho em adicionar tais informações ao modelo. Todavia, também é possível perceber que os resultados de ambos os modelos não são exatamente confiáveis, afinal a acurácia do modelo proposto é somente de 23%.

Esses resultados, embora baixos, possuem uma explicação razoável, principalmente em relação à acurácia: a tarefa simplesmente é muito difícil em alguns casos. Para certas classes em que há 1, 2 ou até 10 casos totais no conjunto de dados de treinamento, não é esperado que um modelo aprenda corretamente a identificar esses casos, afinal de contas, eles são tão raros que não é possível aprender um padrão apenas com uma quantidade tão pequena de exemplos. Não a toa muitos dos artigos que utilizam essa base de dados utilizam apenas 30 classes não tão raras para treinamento e teste.

4 Conclusão

Realizados os testes e uma análise dos modelos é possível concluir algumas coisas a respeito do modelo proposto:

- Primeiramente, pelo resultado de ambos os modelos é evidente que ambas as tarefas avaliadas, com as bases de dados escolhidas, são de alta dificuldade. Afinal, ambos os modelos tiveram resultados decepcionantes nas métricas selecionadas para avaliar os modelos. Assim, trata-se mais de uma comparação entre os dois modelos do que

uma avaliação de quanto esses modelos são bons para serem aplicados em situações reais (principalmente na questão na identificação de anomalias cardíacas, a qual naturalmente exige um grau de confiabilidade maior);

- Em segundo lugar, fica claro que o modelo proposto não é adequado para prever os preços do Bitcoin, pois a adição do mecanismo de auto-atenção baseado no expoente generalizado de Hurst ao N-BEATS piorou o desempenho do modelo em todas as métricas. Uma explicação plausível para isso é a forma como os dados são usados em janelas na previsão enquanto o GHE é calculado sobre janelas maiores da série temporal;

- Já para classificação, o modelo proposto obteve o resultado oposto: uma melhoria em praticamente todos os aspectos em relação ao desempenho do modelo N-BEATS. Destarte, é evidente que a adição das informações do expoente generalizado de Hurst para o modelo auxiliou na classificação dos eletrocardiogramas e, portanto, há indicativos de que o modelo proposto possui vantagens em relação ao N-BEATS clássico quando o assunto é a classificação de séries temporais.

- Pontos a serem melhorados nos modelos incluem a performance geral dos modelos, conseguindo índices melhores de acertos; um estudo mais detalhado sobre se o GHE auxilia apenas nos eletrocardiogramas ou se essa melhoria pode ser esperada em outros casos de classificação; uma análise sobre outras possibilidades de indicadores estatísticos sobre a série temporal que poderiam ser usadas como entradas para o mecanismo de auto-atenção e, idealmente, um algoritmo mais eficiente para calcular o GHE.

Portanto, como conclusão geral, a ideia analisada neste projeto possui mérito e, no caso do problema de classificação, melhora os resultados anteriores. Contudo, o mesmo não ocorre no problema de previsão, onde uma possível solução seria trocar o expoente generalizado de Hurst por algum outro tipo de informação da série temporal mais local, relacionada apenas com os dados pertencentes à janela passada como entrada para o modelo e não uma informação que depende da série inteira.

Referências

[1] Pedro F. G. Bernardinelli and João B. Florindo. Aprendizado profundo na previsão de séries temporais: uma análise comparativa. 2022.

[2] Christopher M. Bishop. *Neural networks for pattern recognition.* Oxford university press, 1995.

[3] Daniel Bourke. Milestone project 3: Time series forecasting in tensorflow.

[4] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley. Physiobank, physiontoolkit and physionet: components of a new reasearch resource for complex physiologic signals. 2000.

[5] Kaiming He, Xiangyu Zhang Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015.

[6] Harold Edwin Hurst. Long-term storage capacity of reservoirs. *Transactions of the American Society of Civil Engineers, Vol. 116, No.1*, 1951.

[7] S. W. Lee and Kim H. Y. Stock market forecasting with super-high dimensional time-series data using convlstm, trend sampling, and specialized data augmentation. *Expert Systems with Applications*, 2020.

[8] T. Di Matteo. Multi-scaling in finance. *Quantitative Finance*, 2007.

[9] Raffaello Morales, T. Di Matteo, Ruggero Gramatica, and Tomaso Aste. Dynamical generalized hurst exponent as a tool to monitor unstable periods in financial time series. *Physica A*, 2012.

[10] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *International Conference on Learning Representations (ICLR)*, 2020.

[11] Bartosz Puszkarski, Krzysztof Hryniów, and Grzegorz Sarwas. N-beats for heart dysfunction classification. 2021.

[12] Bartosz Puszkarski, Krzysztof Hryniów, and Grzegorz Sarwas. Comparison of neural basis expansion analysis for interpretable time series (n-beats) and recurrent neural networks for heart dysfunction classification. *Physiological Measurement*, 43(6):064006, 2022.

[13] Peter Rupprecht. Genhurst.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polushkin. Attention is all you need. *31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

[15] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Wojciech Samek, and Tobias Schaeffter. Ptb-xl, a large publicly available electrocardiography dataset. *PhysioNet*, 2022.

[16] Sebastian Zeng, Florian Graf, Christoph Hofer, and Roland Kwitt. Topological attention for time series forecasting. *35th Conference on Neural Information Processing Systems (NIPS)*, 2021.

[17] Jianwei Zheng, Huimin Chu, Daniele Struppa, Jianming Zhang, Sir Magdi Yacoub, Hesham El-Askary, Anthony Chang, Louis Ehwerhemuepha, Islam Abudayyeh, Alexander Barrett, Guohua Fui, Hai Yao, Dongbo Li, Hangyuan Guo, and Cyril Rakovski. Optimal multi-stage arrythmia classification approach. *Scientific Reports*, 2020.

[18] Jianwei Zheng, Jianming Zhang, Sidy Danioko, Hai Yao, Hangyuan Guo, and Cyril Rakovski. A 12-lead electrocardiogram database for arrythmia research covering more than 10,000 patients. *Scientific Data*, 2020.