Redes Neurais em Processamento de Sinais

COE363 - Telecomunicações

2023.2

Luiz Guilherme de A. Pires
Pedro Eduardo Gonçalves de Carvalho
Lucas Miguel de A. Santos

Relatório referente ao
trabalho da matéria de Telecomunicações

Contents

1 Introdução

Este é o relatório referente ao trabalho prático atribuído à disciplina de Telecomunicações, onde foi visado iniciar um estudo em uma das áreas que vem recebendo grande atenção nos dias de hoje, Machine Learning e Neural Networks, além disso, por em prática questões que aprendemos durante a disciplina. Foi utilizado a linguagem de programação Python dentro de um ambiente do Jupyter notebook, visando uma melhor analise dos dados.

Nosso trabalho foi dividido em três partes:

1. Análise do canto de pássaros
2. Detecção de Anormalidades no Coração

2 Análise do Canto de Pássaros

A princípio, tentaríamos criar uma rede neural capaz de identificar um pássaro a partir de seu canto, mas isso se provou mais complicado na prática em relação à teoria, portanto, ainda na mesma temática, resolvemos optar por um problema mais simples, verificar se uma determinada faixa de áudio é o canto de um pássaro específico.

O pássaro escolhido para o nosso projeto inicial foi Maú(*Perissocephalus tricolor*), um pássaro encontrado ao norte do rio Amazonas. O principal motivo da

Figure 1: Ave Maú
referência: https://www.wikiaves.com.br/wiki/mau

escolha desta ave, foi devido ao seu canto ser bem característicos em comparação com demais pássaros, podendo ser vista na imagem abaixo:

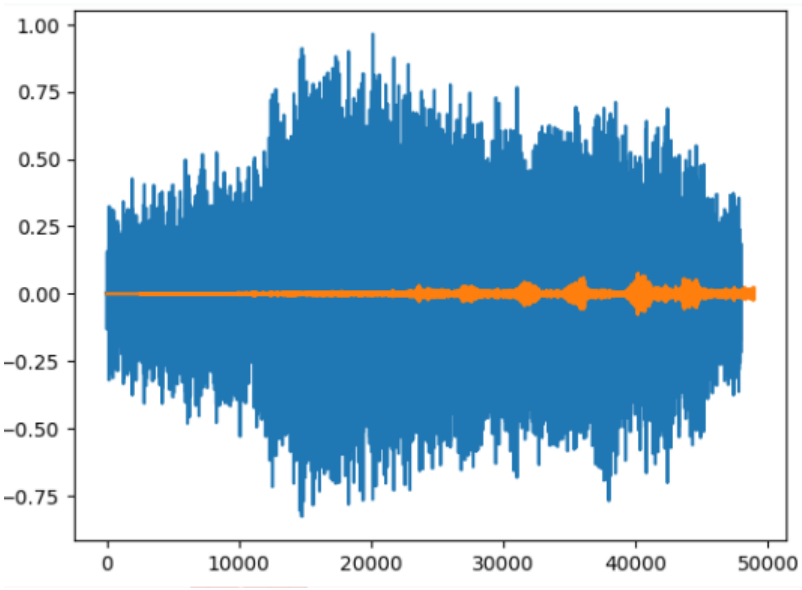

Figure 2: Função de amplitude referente ao canto do maú(azul) e de outro pássaro(laranja)

Pode-se ver que o canto dele é bem distinto, mas ainda sim, não é tão simples para a rede neural analisar este tipo de imagem, para termos uma melhor visualização, utilizamos a chamada Transformada de Fourier de curto termo (STFT), que é o tipo de transformada indicado para funções cujo espectro varia com o tempo, ou seja, não-estacionárias que é o nosso caso.

$$\mathbf{STFT}\{x(t)\} \equiv X(\tau, \omega) = \int_{-\infty}^{\infty} x(t)\omega(t-\tau)e^{-i\omega t}dt$$

Após tendo feito as transformadas obtivemos os seguintes espectrogramas:

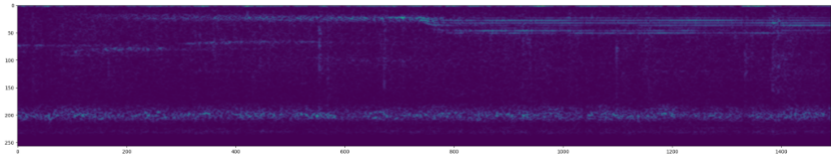

Figure 3: Espectrograma do canto do Maú

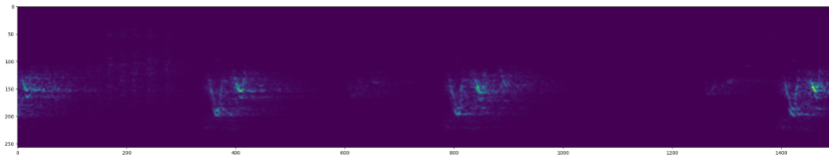

Figure 4: Espectrograma do canto de outros pássaros

Agora que possuímos uma imagem que a rede neural consegue analisar mais facilmente, podemos iniciar a criação da nossa rede e por transformarmos o problema de analise de áudio para um problema de analise de imagem, facilitou o nosso trabalho, tendo em vista que analise de imagem utilizando redes neurais é uma área muito abrangente e com muitos projetos bem sucedidos.

Para o nosso primeiro modelo, utilizamos 3 camadas, e o nosso output será um binário, sendo 1 igual à "é o canto do pássaro Maú" e 0 igual à "Não é o canto do pássaro Maú", ficando desta forma:

1. Input(Espectrograma)
2. Camada Convolucional 2d
3. Camada *Flatten*
4. Camada Densa
5. Output(Binário)

Utilizando matrizes 3x3 como filtro da camanda convolucional, funções de ativação do tipo "RELU"(rectified linear unit) e por fim, na camada densa uma função de ativação do tipo "sigmoid" para a representação do binário, utilizamos também o algoritmo de otimização chamado Adam e treinamos a nossa rede por 4 *epochs*, obtivemos bons resultados, entretanto podíamos fazer melhor, assim, dobramos a quantidade de camada densa e convolucional, ficando desta forma:

1. Input(Espectrograma)
2. Camada Convolucional 2d
3. Camada Convolucional 2d
4. Camada *Flatten*
5. Camada Densa
6. Camada Densa
7. Output(Binário)

Utilizando os mesmos detalhes do modelo anterior e 4 *epochs* de treino, obtivemos o resultado de 100% de acurácia, finalizando com sucesso nossa primeira parte do trabalho

2.1 Link para o Código

- https://github.com/pedroedut7/Telecom_Final/blob/main/PassaroClassifica%20(1).ipynb

3 Detecção de Anormalidades no Coração

Com o amplo uso de redes neurais convolucionais (CNN) em muitos campos, a CNN também se tornou um recurso popular opção para classificar automaticamente os sinais de ECG (eletrocardiogramas) registrados pelo Holter(é um tipo de exame que reproduz graficamente a atividade elétrica do coração em funcionamento). Comparado com os métodos tradicionais, a CNN o classificador pode inserir pulsações diretamente sem extração e seleção de recursos adicionais; também demonstra competitividade na precisão da classificação.

Com isso, para nossa segunda parte do trabalho, diferente, da primeira parte onde utilizamos camadas convolucionais de duas dimensões, para essa etapa utilizaremos camadas com 1 dimensão para fazer uma analise sobre a normalidade dos batimentos cardíacos dado os sinais de um ECG, segue abaixo exemplos dos sinais utilizados para o treino da rede:

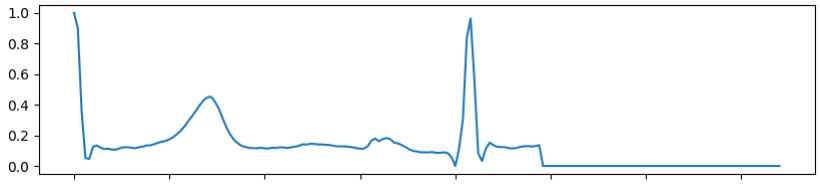

Figure 5: ECG de um coração normal

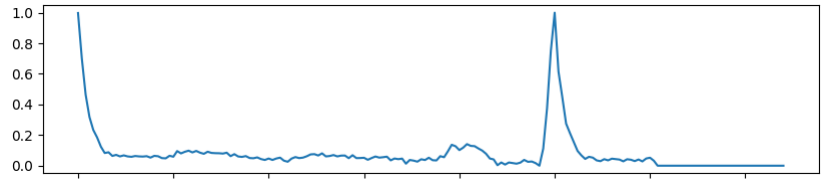

Figure 6: ECG de um coração anormal

Nós pegamos uma base da mesma fonte da primeira parte do trabalho, o Kaggle, tendo as seguintes características:

- Número de amostras: 14552
- Número de categorias: 2(normal e anormal)
- Frequencia de amostragem: 125Hz

- Fonte dos dados: *Physionet's PTB Diagnostic Database*

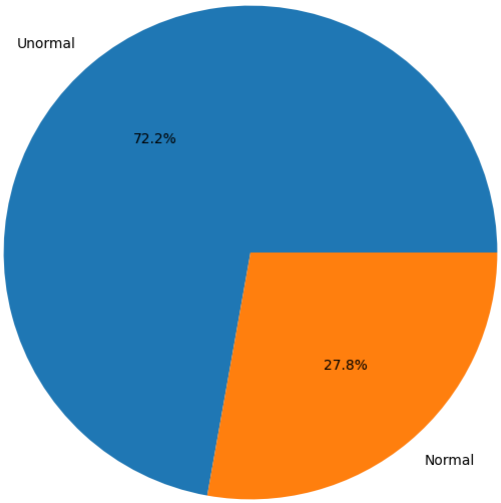

Figure 7: Gráfico de pizza das classes

Como os dados que utilizamos já estavam na formatação correta, não precisamos realizar um pre-processamento igual fizemos na primeira parte do projeto, assim, primeiramente tentamos utilizar o mesmo modelo da parte anterior, mudando apenas a quantidade de dimensões utilizadas na camada convolucional, tendo em vista que os *inputs* e *outputs* eram similares. O resultado que obtivemos foi que o código acertava meramente por sorte, algo que não pode ocorrer numa analise de um eletrocardiograma.

Após algumas pesquisas, verificamos que apenas duas camadas convolucionais para um problema complexo como esse não seria o suficiente, assim, como no primeiro modelo, dobramos a quantidade de camadas convolucionais e aumentamos em uma camada densa, o que gerou um aumento na acurácia, contudo, o nosso valor de *Recall*(porcentagem de positivos reais que foram classificados corretamente = verdairos positivos/(verdadeiros positivos + falsos negativos)) estava baixo.

Depois de mais algumas pesquisas em trabalhos similares, vimos que muitos utilizavam após a camada convolucional funções chamadas *BatchNormalization* e *MaxPooling1D*, sendo a primeira como o nome implica, realiza a normalização dos *inputs* mantendo a média próxima de zero e o desvio padrão de um, já o segundo realiza um *downsample* na representação dos *inputs* pegando o valor máximo em uma determinada janela de tempo (por padrão 2).

Rodando os testes, verificamos que o valor de *Recall* ainda estava menor do que o valor da precisão, indicando que existia uma quantia considerável de falsos negativos, algo que não pode ocorrer, assim, fizemos mais testes mudando a

função de ativação de algumas camadas, que normalmente é a RELU, trocamos para outro tipo de ativação, a SELU(Scaled Exponential Linear Unit) percebemos uma melhora no valor de *Recall*, mas somente quando a função SELU é utilizada na camada convoulucional antes da camada *flatten*, faz com que tanto a precisão quanto o *Recall* fossem maiores, obtendo cerca de 92% de precisão e valor de *Recall* igual à 0.9852 ao final da última *epoch*, assim, ficamos com o seguinte modelo:

- Inputs
- Camada Convolucional 1D
- BatchNormalization
- MaxPooling1D
- Camada Convolucional 1D
- BatchNormalization
- MaxPooling1D
- Camada Convolucional 1D
- BatchNormalization
- Camada Convolucional 1D (com ativação SELU)
- GlobalAveragePooling1D
- Camada *Flatten*
- Camada Densa
- Dropout (utilizado para diminuir o *overfitting* do modelo)
- Camada Densa
- Camada Densa (output)

Utilizamos o mesmo algoritmo de otimização, mas como esse era um trabalho de complexidade maior, fizemos uso de mais *epochs*, sendo 100 *epochs* para treinar a rede neural.

3.1 Link para o Código

- https://github.com/pedroedut7/ECG_Classification/blob/main/ECG.ipynb

4 Referências

- https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8952723
- https://arxiv.org/pdf/1707.01836.pdf
- https://arxiv.org/pdf/2002.00254.pdf
- https://arxiv.org/pdf/1804.06812.pdf
- https://keras.io
- https://www.tensorflow.org/?hl=pt-br
- *Datasets*: https://www.kaggle.com