Reconhecimento de choro de bebês

1. Introdução

O relatório em questão, feito pelos componentes Leandro Rodrigues e Tobias Aguiar, visa utilizar um modelo de aprendizagem de máquina para realizar um reconhecedor de choro de bebês. É de conhecimento geral de que as mães normalmente possuem um ouvido mais sensível para reconhecer quando alguma criança de sua família está chorando. Mas nem sempre essa “audição especial” funciona e em muitos dos casos não estão em casa e as outras pessoas que estão nem sempre se atentam a este fato, onde as necessidades de uma criança podem variar de fome, sono até mesmo o desconforto. Logo, é necessário uma garantia de suporte a todos esses indivíduos, pois são frágeis e são sujeitos facilmentes à perigos. Além disso, sempre que os bebês sentem que estão em situação de perigo, tendem a chorar para buscar ajuda.

Para tentar dar um suporte a eles sem a dependência da audição sensível da mãe, foi desenvolvido um código reconhecedor de choro de bebês, utilizando-se um modelo de aprendizagem de máquina, podendo ser ampliado e instalado e embarcado junto ao *software* algum tipo de alarme que possa alertar quem está o acompanhando para poder prestar apoio.

2.Metodologia

2.1 Processamento de sinais

Um sinal analógico é um processo físico que depende do tempo e pode ser modelado por uma função real sobre uma variável real t que representa o tempo. No caso de sinais de áudio é comum que esta função representa a pressão sonora, o valor da corrente elétrica que corre por um fio ou a posição da membrana de um alto falante num momento específico. É raro que um sinal analógico possa ser representado por uma expressão analítica, pois a maioria dos sinais analógicos são muito complexos e possuem ruído.

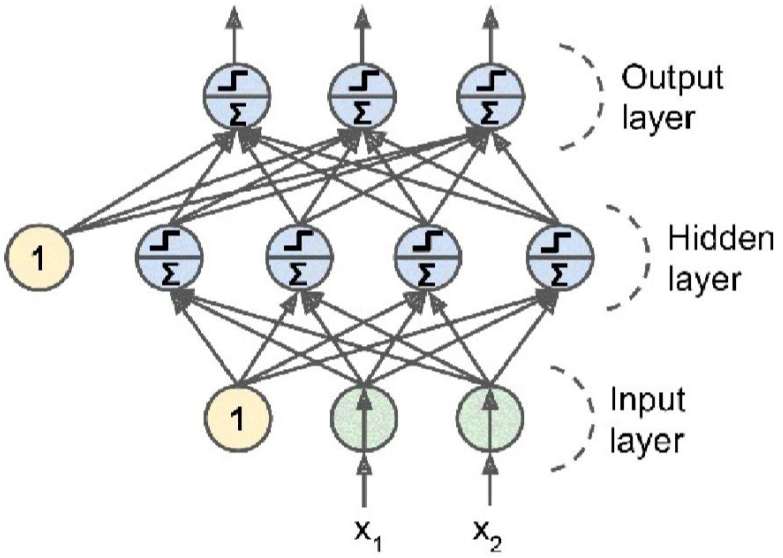
Um sinal analógico pode ser digitalizado por um processo de amostragem, que aproxima valores do sinal analógico tomados em intervalos igualmente espaçados no tempo. Dado um sinal analógico x(t) definido em um intervalo $a \leq t < b$ e um inteiro N que representa a quantidade de amostras a serem obtidas do sinal analógico, o intervalo de amostragem é definido como $\Delta t := (b - a)/N$.

Um sinal digital periódico de período N pode ser completamente descrito por N componentes senoidais com frequências harmonicamente relacionadas. A Transformada de Fourier Discreta (em inglês Discrete Fourier Transform, ou DFT) é uma função sobre um vetor de tamanho N que corresponde a uma mudança de base em um determinado espaço vetorial e permite o cálculo das N componentes de frequência do sinal. A Transformada Rápida de Fourier (em inglês Fast Fourier Transform, ou FFT) é uma implementação eficiente da DFT que diminui sua complexidade de O(N2) para O(N log(N)), onde N é o número de amostras no domínio do tempo ou, de forma equivalente, o número de índices de frequência que descrevem o espectro do sinal após a computação da Transformada.

2.2 Multilayer Perceptron (MLP)

Uma MLP é composta por uma camada de entrada (passagem), uma ou mais camadas de LTUs, chamadas camadas ocultas, e uma camada final de LTUs, chamada camada de saída (veja a Figura 01). Cada camada, exceto a camada de saída, inclui um neurônio de polarização e está totalmente conectado à próxima camada. Quando uma RNA tem duas ou mais camadas ocultas, é chamada de rede neural profunda (DNN).

Figura 01- Multilayer Perceptrons



Para cada instância de treinamento, o algoritmo o alimenta na rede e calcula a saída de cada neurônio em cada camada consecutiva (essa é a passagem direta, exatamente como ao fazer previsões). Em seguida, mede o erro de saída da rede (ou seja, a diferença entre a saída desejada e a saída real da rede) e calcula quanto cada neurônio na última camada oculta contribuiu para o erro de cada neurônio de saída. Em seguida, ele mede o quanto dessas contribuições de erro vieram de cada neurônio na camada oculta anterior - e assim por diante até o algoritmo atingir a camada de entrada. Essa passagem reversa mede com eficiência o gradiente de erro em todos os pesos de conexão na rede, propagando o gradiente de erro para trás na rede (daí o nome do algoritmo).. A última etapa do algoritmo de retropropagação é uma etapa de Descida de gradiente em todos os pesos de conexão na rede, usando os gradientes de erro medidos anteriormente.

Resumindo: para cada instância de treinamento, o algoritmo de retropropagação primeiro faz uma previsão (passagem direta), mede o erro e passa por cada camada no

sentido inverso para medir a contribuição do erro de cada conexão (passagem reversa) e, finalmente, ligeiramente os pesos da conexão para reduzir o erro (etapa de descida do gradiente). Portanto, uma MLP é frequentemente usado para classificação, com cada saída correspondendo a uma classe binária diferente (por exemplo: urgente / não urgente etc.)

3.Principais partes do código

O código de reconhecimento de choro se baseia no processamento de áudio em conjunto com o método de machine learning MLP. A variável que representa um áudio é um vetor obtido da função “read”, tal vetor é usado para obter o espectro. No nosso caso, usamos o espectro como vetor de entrada para o multilayer. Todo esse desenvolvimento pode ser visto na figura 01 e 02.

Figura 01. Processamento digital de voz

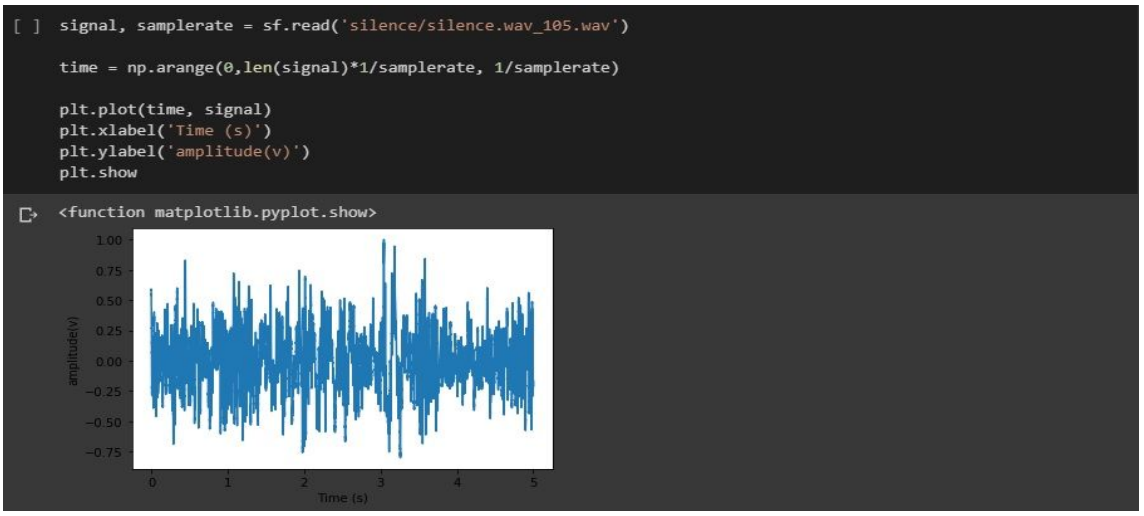
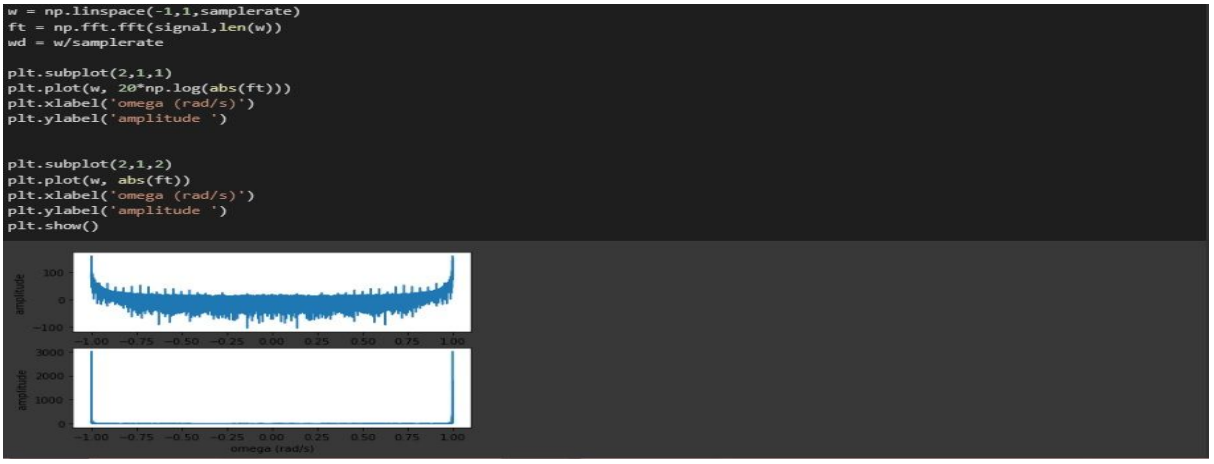


Figura 02. Espectro do sinal



O método MLP necessita de entradas e saídas(binárias). Neste caso, o vetor de espectros já obtido serviu de entrada e saída são os labels(choro/sorriso, choro/barulho etc). Assim, obteve o percentual de treino e teste(figura 03). O treino foi caracterizado com os parâmetros de camada inicial, oculta e final visto na figura 04. Por último, com a rede pronto, é possível fazer a previsão com os dados de testes(figura 05).

Figura 03- Método MLP

```
[ ]  # Splitting the dataset into the Training set and Test set
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(audios, labels_enc, test_size = 0.25, random_state = 0)

[ ]  # Feature Scaling
     from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)
```

Figura 04- Características da rede

```
[ ]  # Part 2 - Now let's make the ANN!

     # Importing the Keras libraries and packages
     import keras
     from keras.models import Sequential
     from keras.layers import Dense

     # Initialising the ANN
     classifier = Sequential()

     # Adding the input layer and the first hidden layer
     classifier.add(Dense( activation = 'relu', input_dim = 220500, units = 8, kernel_initializer = 'uniform'))

     # Adding the second hidden layer
     classifier.add(Dense( activation = 'relu', units = 12, kernel_initializer = 'uniform' ))


     # Adding the output layer
     classifier.add(Dense( activation = 'sigmoid', units = 1, kernel_initializer = 'uniform'))

     # Compiling the ANN
     classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

[ ]  # Fitting the ANN to the Training set
     classifier.fit(X_train, y_train, batch_size = 8, epochs = 50)
```

Figura 05 - Obtenção da previsão

```
[ ]  # Part 3 - Making the predictions and evaluating the model

     # Predicting the Test set results
     y_pred = classifier.predict(X_test)
     print(y_pred[0:10])

     y_pred = (y_pred > 0.5)
     print(y_pred[0:10])

     # Making the Confusion Matrix
     from sklearn.metrics import confusion_matrix
     cm = confusion_matrix(y_test, y_pred)
```

4.Experimentos

A base de dados usada tinha áudios de choro, silêncio, barulho e riso e o ideal era que todos esses dados fossem testado juntos e o código pudesse identificar, porém a técnica de multilayer perceptron só pode classificar de forma binária. Então, foi testado de 2 em 2: choro/silêncio e choro/riso; o resultado pode ser visto na figura 06. A matriz de confusão mostra que o 100% de acerto.

Esse resultado só foi possível porque as frequência entre as classes são bastante distintas. Por exemplo, o silêncio e o choro poderia ser diferenciado apenas pela energia. Conclui-se que o ideal a ser usado no reconhecimento de choro era o método de redes neurais e um estudo mais aprofundado dos métodos de processamento de voz, pois no caso de latido de cachorro e choro de bebê, só dados de frequência não é necessário.

Figura 06- Resultados obtidos

Código: https://drive.google.com/open?id=1-w1HShviT-zcwB_4jVFiTt9Y7vHndzJW