

Faculdade de Ciências e Tecnologias

ANÁLISE E TRANSFORMAÇÃO DE DADOS - LICENCIATURA EM

ENGENHARIA INFORMÁTICA

**Identificação de Dígitos Através de Características Extraídas de Sinais de Áudio**

Trabalho realizado por:

Teodoro Marques e Bernardo Direito

Março, 2025

META 1

### **1/2 - Criação da estrutura de dados e importação dos sinais de áudio**

# O processo de importação dos sinais de áudio inicia-se com dois *for* loops que percorrem todas as combinações possíveis entre os dígitos e as repetições. Para cada iteração, é utilizada a função *sprintf*, a qual obtém o dígito, o ID do participante (constante e igual a 40) e o número da repetição, e cria o caminho (path) completo do ficheiro com a função *fullfile*, garantindo, por sua vez, a correta formatação do caminho. Os dados referentes a cada gravação, como o nome e o caminho do arquivo, o valor do dígito, o número da repetição e o identificador do participante, são então armazenados nos respectivos vetores e ­cells.

# De seguida, o ficheiro de áudio é lido através da função *audioread*. Caso a leitura encontre algum problema – por exemplo, se o arquivo não existir – o código captura(dá *catch*) a exceção, emite um aviso e atribui um sinal vazio e uma taxa de amostragem indefinida (*NaN*) à gravação em questão, garantindo assim que o processo continue sem interrupções. Após a leitura de cada arquivo, o sinal de áudio e a taxa de amostragem são armazenados nas variáveis correspondentes.

# Por fim, todos os dados retraídos são postos em uma tabela utilizando a função *table*. Essa tabela organiza os metadados e os sinais de áudio em colunas nomeadas, permitindo um acesso mais fácil e uma análise integrada dos dados.

### **3 - Representação gráfica de um exemplo dos sinais importados**

É criado um plot para exibir os gráficos dos sinais de áudio originais, utilizando um *for* que percorre os dígitos de 0 a 8. Para cada dígito, ele extrai da tabela de metadados o índice de uma gravação específica correspondente à repetição número 5 (neste exemplo) e, caso o sinal exista, recupera os dados do sinal e a taxa de amostragem para construir um eixo temporal. Em seguida, o sinal é representado em um subplot individual, com rótulos adequados nos eixos, título que indica o dígito e a repetição, e ajustes na amplitude do eixo vertical que variam conforme o dígito apresentado. Se o sinal não for encontrado, o subplot exibe uma mensagem indicando a ausência do dado para aquele dígito. No final, a figura recebe um título geral, reunindo todos os gráficos dos sinais de áudio.

### **4 - Pré-processamento dos ficheiros**

### O objetivo desta etapa é padronizar tanto a duração quanto o intervalo de amplitude dos sinais. Para isso, cria-se uma nova estrutura de dados, onde cada gravação passa por um processo de remoção do silêncio inicial, baseado em um limiar de energia, e é normalizada para que seus valores de amplitude se situem entre -1 e 1. Essa etapa é realizada por meio de uma função específica *chamada removeInitialSilenceAndNormalize*, aplicada a cada sinal de áudio presente na tabela de metadados. Se o sinal estiver vazio, o código preserva essa condição. Após o processamento de todas as gravações, os sinais pré-processados são armazenados em uma nova coluna da tabela, *denominada PreprocessedAudioSignal*, integrando os dados tratados à estrutura geral dos metadados.

### **5 – Representação Gráfica dos Sinais de Áudio Pré-processados (Dígitos 0–9, Repetição 5)**

Após o pré-processamento dos sinais de áudio descrito no ponto 4, que incluiu tanto a remoção do silêncio inicial com base num limiar de energia como a normalização da amplitude para o intervalo [-1, 1], foi gerada uma representação gráfica dos sinais pré-processados correspondentes aos dígitos de 0 a 9, especificamente para a repetição número 5. Esta etapa foi implementada no script *AUDIO\_DIGIT\_ANALYSIS.M* através da criação de uma nova figura intitulada "Preprocessed Audio Signal Plots". Um ciclo *for* percorre os dígitos de 0 a 8, organizando os gráficos numa grelha de 3x3 subplots, replicando o formato da Figura 2 do projeto, embora o dígito 9 não seja plotado devido à limitação do layout 3x3 (os seus dados, no entanto, permanecem na tabela de metadados).

Para cada dígito, o código identifica o índice na tabela *metadataTable* onde *DigitValue* corresponde ao dígito atual e *RepetitionNumber* é 5. Se o sinal pré-processado (*PreprocessedAudioSignal*) existir e não for vazio, os dados do sinal e a taxa de amostragem (*SamplingRate*) são recuperados. O eixo temporal é calculado como *(0:length(signalData) - 1) / sampleRate*, expresso em segundos, e o sinal é “plotado” com uma linha azul contínua, utilizando a função *plot*. Os eixos são rotulados como "*Time (s)*" (horizontal) e "*Amplitude*" (vertical), com um título específico para cada subplot no formato "*Digit X, Repetition 5*". O intervalo do eixo vertical é fixado entre -1 e 1, refletindo a normalização realizada, e uma grelha é adicionada para facilitar a visualização. Caso o sinal esteja ausente ou vazio, o subplot exibe a mensagem "*Digit X (not found)*". Um título para a figura, "*Preprocessed Audio Signal Plots*", é aplicado com a função *sgtitle.* Assim, conseguimos observar com clareza os sinais após o pré-processamento, que destaca a remoção do silêncio inicial e a uniformização da amplitude, o que acaba por facilitar a análise visual das formas de onda e padrões temporais característicos de cada dígito.

### **6 - Comparação Visual dos Gráficos Obtidos nos Pontos 3 e 5**

A comparação visual entre os gráficos dos sinais originais (ponto 3, figura "*Original Audio Signal Plots*") e os sinais pré-processados (ponto 5, figura "*Preprocessed Audio Signal Plots*") foi realizada de forma qualitativa, conforme solicitado no projeto. Nos gráficos originais, os sinais brutos dos dígitos 0 a 8 (repetição 5) apresentam amplitudes variáveis, ajustadas manualmente para cada dígito (por exemplo, [-0.02, 0.02] para os dígitos 0, 2 e 8; [-0.03, 0.03] para 1, 3, 4, 5 e 7; e [-0.04, 0.04] para 6) de forma a termos melhor visualização dos sinais, e frequentemente incluem períodos de silêncio inicial ou ruído de fundo, dependendo das condições de gravação. Estas variações dificultam a identificação direta de padrões temporais consistentes entre os dígitos.

Já nos gráficos pré-processados, os sinais exibem uma padronização significativa: o silêncio inicial foi removido utilizando a função *removeInitialSilenceAndNormalize*, que aplica um limiar de energia de 0.01, e a amplitude foi normalizada para o intervalo [-1, 1], eliminando diferenças decorrentes de fatores como a distância ao microfone. Visualmente, os sinais pré-processados começam no momento exato em que o som relevante é detetado, apresentando formas de onda mais claras e uniformes em termos de amplitude. Esta transformação destaca características temporais específicas, como a duração dos sinais, os picos de amplitude e os padrões de oscilação, que se tornam mais evidentes. Por exemplo, dígitos como o 0 podem exibir formas mais suaves e prolongadas, enquanto o 1 pode apresentar picos mais abruptos, embora estas observações dependam da análise detalhada de cada sinal.

A comparação sugere que o pré-processamento melhora a consistência visual e a capacidade de diferenciação entre os dígitos, ao remover ruído irrelevante e padronizar os sinais, preparando-os para a extração de características quantitativas nas etapas seguintes.

### **7 - Implementação do Cálculo de Características Temporais**

Para possibilitar a diferenciação entre os dígitos com base em propriedades temporais, foram calculadas cinco características para cada sinal pré-processado, implementadas no script através da função auxiliar *extractTemporalFeatures*. Estas características foram aplicadas a todas as 500 gravações (50 repetições por cada um dos 10 dígitos) e incluem:

**Energia Total**: Calculada como a soma dos quadrados das amostras do sinal (sum(signal .^ 2)), representando a intensidade global do som.

**Desvio Padrão**: Obtido com a função *std(signal)*, indicando a variabilidade da amplitude ao longo do tempo.

**Amplitude Máxima**: Determinada como o valor absoluto máximo do sinal (*max(abs(signal))*), refletindo o pico de intensidade.

**Taxa de Passagem por Zero**: Calculada pela função *calculateZeroCrossingRate*, que conta o número de vezes que o sinal cruza o valor zero, dividido pelo número total de amostras menos um (*sum(signDifference ~= 0) / (length(signal) - 1)*), indicando a frequência das mudanças de polaridade.

**Duração**: Calculada como o número de amostras dividido pela taxa de amostragem (*length(signal) / sampleRate*), representando o tempo total do sinal em segundos.

O cálculo foi realizado iterativamente para cada entrada da coluna *PreprocessedAudioSignal* da tabela *metadataTable*. Se o sinal estivesse vazio, valores nulos (zeros) eram atribuídos às características correspondentes, garantindo a continuidade do processo. Os resultados foram armazenados em novas colunas da tabela: *TotalEnergy*, *StandardDeviation*, *MaximumAmplitude*, *ZeroCrossingRate* e *Duration*. Esta abordagem fornece uma base quantitativa para caracterizar os sinais, nos ajudando na análise de padrões temporais que diferenciem os dígitos nas etapas subsequentes.

### **8 - Identificação das Três Melhores Características para Discriminação dos Dígitos**

Este ponto visa mostrar a visualização gráfica de cada feature. Inicia-se a gerar um boxplot que exibe a distribuição da energia total de cada dígito. Para isso, uma figura é criada e um boxchart é utilizado para representar a variável de energia total agrupada por dígito, onde os dígitos são convertidos em categorias para facilitar a comparação visual. De seguida, produz-se um gráfico de dispersão tridimensional que relaciona as características (a taxa de cruzamento por zero, o desvio padrão e a duração dos sinais). Nesse gráfico, cada dígito (de 0 a 9) é representado por uma cor específica, permitindo identificar a dispersão das características dos sinais em um espaço 3D, com eixos que correspondem a cada uma das variáveis mencionadas.

Posteriormente, criamos uma série de gráficos de dispersão para os recursos extraídos, como energia total, desvio padrão, amplitude máxima, taxa de cruzamento por zero e duração. Para cada característica, são criados subplots que apresentam a variação dos valores dessa característica em função dos dígitos, permitindo observar as diferenças entre os sinais conforme o dígito. Cada subplot é rotulado com o nome da característica, e os pontos correspondentes a cada dígito são “plotados” em diferentes posições para facilitar a comparação.

Por fim, o script conclui a análise visual criando boxplots para cada uma das características extraídas. Esses boxplots utilizam os dígitos como categorias no eixo horizontal e os valores da respetiva característica no eixo vertical, oferecendo uma visão estatística sobre a distribuição e a variabilidade dos dados para cada dígito. Dessa forma, o conjunto de gráficos elaborado não só demonstra a distribuição dos sinais em termos de energia total, mas também permite a análise comparativa de múltiplos atributos dos sinais, facilitando a identificação de padrões e a compreensão das características extraídas em relação a cada dígito.

### **9 - Remoção dos Sinais de Áudio da Estrutura de Dados e Salvamento**

Por fim, a estrutura de dados é preparada para armazenamento e otimização. As colunas que continham os sinais de áudio originais e os sinais pré-processados são retiradas, substituindo os dados por vetores vazios (como indicado abaixo):

***metadataTable.RawAudioSignal = [];***

***metadataTable.PreprocessedAudioSignal = [];***

O que pode ser feito para economizar espaço ou garantir que os dados sensíveis não sejam mantidos desnecessariamente. Em seguida, a tabela completa, agora contendo apenas os metadados essenciais, é guardada em um arquivo MAT intitulado "*finalAudioData.mat*", com a função *save:*

***save('finalAudioData.mat', 'metadataTable');***