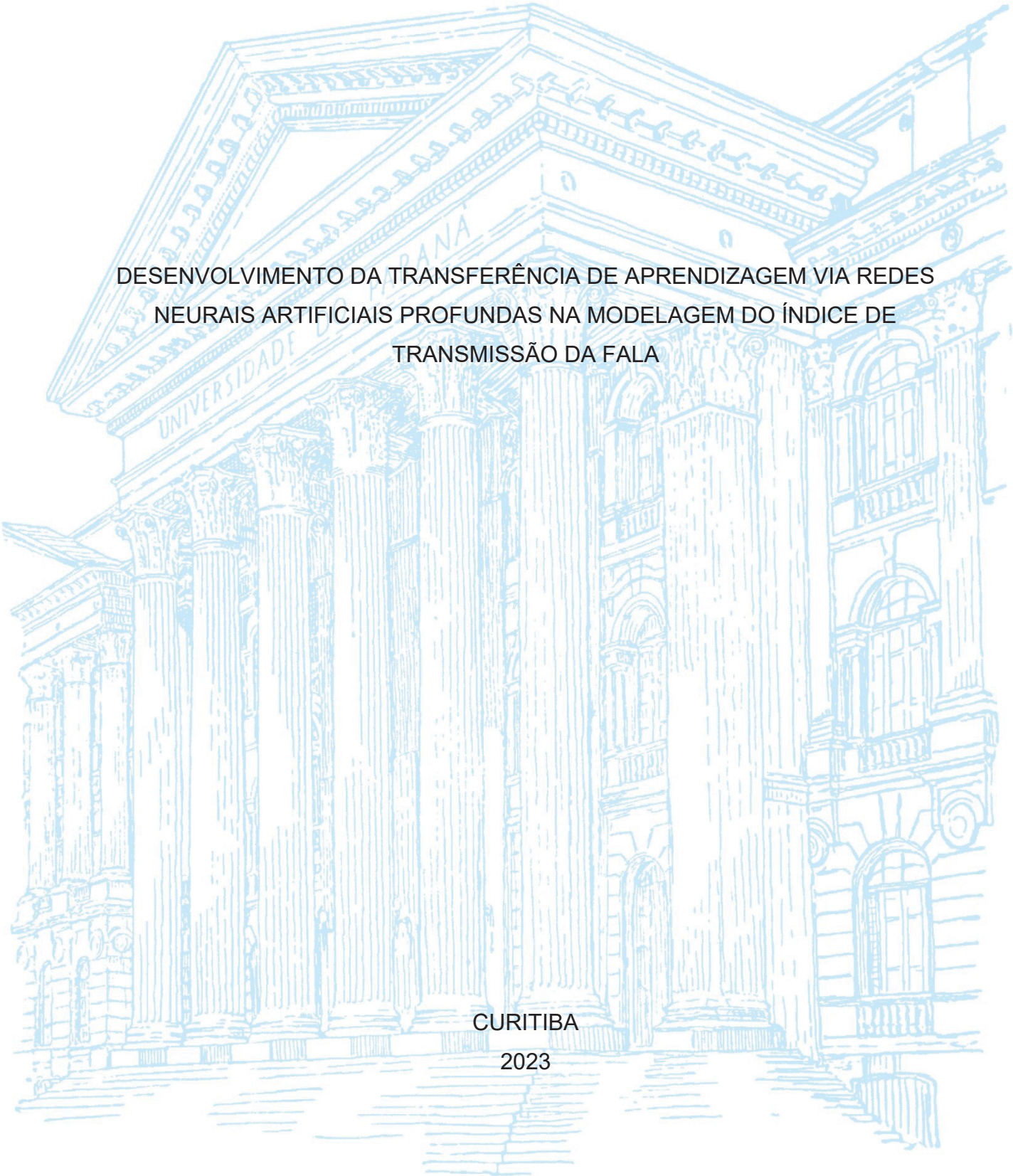
UNIVERSIDADE FEDERAL DO PARANÁ

ERIBERTO OLIVEIRA DO NASCIMENTO



# DESENVOLVIMENTO DA TRANSFERÊNCIA DE APRENDIZAGEM VIA REDES NEURAIS ARTIFICIAIS PROFUNDAS NA MODELAGEM DO ÍNDICE DE TRANSMISSÃO DA FALA

CURITIBA

2023

ERIBERTO OLIVEIRA DO NASCIMENTO

# DESENVOLVIMENTO DA TRANSFERÊNCIA DE APRENDIZAGEM VIA REDES NEURAIS ARTIFICIAIS PROFUNDAS NA MODELAGEM DO ÍNDICE DE TRANSMISSÃO DA FALA

Tese apresentada ao programa de Pós-Graduação em Engenharia Mecânica, Setor de Tecnologia, da Universidade Federal do Paraná, como requisito parcial à obtenção do título de Doutor em Engenharia Mecânica na área de concentração de Fenômenos de Transporte e Mecânica dos Sólidos.

Orientador: Prof. Titular Dr. -Ing. Paulo Henrique Trombetta Zannin

CURITIBA

2023

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA CIÊNCIA E TECNOLOGIA

Nascimento, Eriberto Oliveira do
Desenvolvimento da transferência de aprendizagem via redes neurais artificiais profundas na modelagem do índice de transmissão da fala. / Eriberto Oliveira do Nascimento. – Curitiba, 2023.
1 recurso on-line : PDF.

Tese (Doutorado) - Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia Mecânica.

Orientador: Prof. Dr. -Ing. Paulo Henrique Trombetta Zannin

1. Acústica. 2. Redes neurais convolucionais. 3. Inteligência artificial generativa. I. Zannin, Paulo Henrique Trombetta. II. Universidade Federal do Paraná. Programa de Pós-Graduação em Engenharia Mecânica. III. Título.

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585

MINISTÉRIO DA EDUCAÇÃO
SETOR DE TECNOLOGIA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO ENGENHARIA MECÂNICA - 40001016040P5

# TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação ENGENHARIA MECÂNICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **ERIBERTO OLIVEIRA DO NASCIMENTO** intitulada: **DESENVOLVIMENTO DA TRANSFERENCIA DE APRENDIZAGEM VIA REDES NEURAIS ARTIFICIAIS PROFUNDAS NA MODELAGEM DO INDICE DE TRANSMISSAO DA FALA**, sob orientação do Prof. Dr. PAULO HENRIQUE TROMBETTA ZANNIN, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutor está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 14 de Junho de 2023.

Assinatura Eletrônica
17/06/2023 13:19:21.0
PAULO HENRIQUE TROMBETTA ZANNIN
Presidente da Banca Examinadora

Assinatura Eletrônica
09/11/2023 15:14:59.0
ARCANJO LENZI
Avaliador Externo (UNIVERSIDADE FEDERAL DE SANTA CATARINA)

Assinatura Eletrônica
09/11/2023 08:47:41.0
NILSON BARBIERI
Avaliador Externo (PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ)

Assinatura Eletrônica
16/06/2023 10:25:51.0
EDUARDO MÁRCIO DE OLIVEIRA LOPES
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

## AGRADECIMENTOS

Ao meu orientador, Prof. Tit. Dr.-Ing. Paulo H. T. Zannin, pelos valiosos ensinamentos e orientação, tanto durante o mestrado, quanto no doutorado. Por acreditar na proposta de trabalho e sempre incentivar um alto nível de comprometimento com as pesquisas. Agradeço, também, por abdicar de sábados com a família para realizar as medições acústicas.

Ao Prof. Eduardo M. O. Lopes, pelas aulas, conselhos e ensinamentos. O exemplo da postura em comprometimento com a qualidade, e a lisura serão sempre guardados comigo.

Ao Prof. Lucas Nonato de Oliveira, e à Profa. Linda Viola Ehlin Caldas, que desde a iniciação científica estiveram comigo nessa jornada acadêmica. Ao Lucas pela amizade de tantos longos anos. Assim como para o Prof. M. Felipe Luz, pela amizade, suporte e por ter me recebido em Curitiba, assim como toda a sua família.

Aos inúmeros colegas de laboratório desde 2017. Aqui cito apenas alguns nomes: Júlio Herrmann (por toda a ajuda durante as medições e por ter passado sua experiência sobre o STI), aos colegas Gabriel Pértile, Rafael Ferraz, Giovanne Lima, Thomas Jeferson, Daniel Souza, Matheus Mazur, Gabrielle Schittini, Carla Dechechi, Lígia Medina e Caroline Amorim.

Estendo meus agradecimentos ao secretário do PGMEC, Jonatas Ricardo Zanoto, por todo o auxílio e ótimas conversas.

Para o PGMEC e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES, pela bolsa de estudos.

A Universidade Federal do Paraná (UFPR), pela ótima estrutura, e ao Prof. Zannin, pela ótima estrutura do Laboratório de Acústica Ambiental, Industrial e Conforto Acústico – LAAICA, que possui umas das melhores estruturas em termos de pessoal e instrumentação para pesquisar e estudar acústica no Brasil.

“A good many times I have been present at gatherings of people who, by the standards of the traditional culture, are thought highly educated and who have with considerable gusto been expressing their incredulity of scientists. Once or twice, I have been provoked and have asked the company how many of them could describe the Second Law of Thermodynamics. The response was cold: it was also negative. Yet I was asking something which is the scientific equivalent of: Have you read a work of Shakespeare's? I now believe that if I had asked an even simpler question -- such as, What do you mean by mass, or acceleration, which is the scientific equivalent of saying, Can you read? -- not more than one in ten of the highly educated would have felt that I was speaking the same language. So the great edifice of modern physics goes up, and the majority of the cleverest people in the western world have about as much insight into it as their neolithic ancestors would have had.”

— C.P. Snow

SNOW, Charles Percy. **The Two Cultures and the Scientific Revolution**. (Repr.). Cambridge [Eng.]: University Press, 1959.

## RESUMO

O conforto acústico em salas de aula interfere na dinâmica do ensino-aprendizagem. Assim, salas com problemas acústicos causam um ônus socioeconômico, comprometendo a eficiência da educação. A inteligibilidade da fala é mensurada pelo Índice de Transmissão da Fala, em inglês, *Speech Transmission Index* (STI). Contudo, a medição desse índice é complexa e cara. Na literatura corrente, há modelos preditivos do STI que usam o Tempo de Reverberação (TR) como variável de regressão. Entretanto, esses modelos não ponderam os efeitos espectrais da localização dual tempo-frequência do sinal da resposta impulsiva das salas (RIR), bem como o conteúdo espectral do ruído de fundo. Este trabalho objetivou gerar um modelo preditivo do STI ao aplicar uma rede neural convolucional-unidimensional profunda (1DCONVNET). Contudo, devido à dificuldade em realizar medições *in situ* do STI, e com o objetivo de prover robustez às estimativas da rede 1DCONVNET, foi proposto um novo algoritmo de Transferência de Aprendizagem (TF), denominado de Minimização Variacional Projetiva-Adaptativa não Paramétrica (MVPAnP). O MVPAnP objetiva otimizar um modelo generativo treinado com dados simulados e transferir a generalização para um conjunto de dados de medições. O algoritmo MVPAnP baseia-se em Inteligência Artificial Generativa, via Redes Autocodificadoras Variacionais (VAE). A rede VAE gerou uma distribuição à posteriori agregada no espaço latente de origem que foi aplicada como parametrização da distribuição do conjunto alvo. A relação entre os conjuntos de origem e alvo deu-se via regressão de componentes principais de núcleo (KPCR) sobre os espaços latentes gerados pela rede VAE. A dimensão dos conjuntos de treinamento e validação da rede VAE foi de 25000 e 5000 salas, respectivamente. A rede 1DCONVNET possui dois sinais de entrada, a resposta impulsiva da sala, simulada via o Método das Imagens e o ruído de fundo (BGN), ambas entradas foram processadas pela Densidade Espectral de Potência (PSD), a saída foi o STI. A validação experimental da rede 1DCONVNET foi realizada em 13 salas de aula, sendo 301 de medições de TR - ISO 3288-2 e 90 medições do STI - IEC 60268-16. Dessas medições gerou-se o conjunto de testes com 600 salas, este conjunto foi utilizado para avaliar a rede 1DCONVNET nos cenários com e sem TF, tal configuração dos cenários é representada pelo acoplamento do algoritmo MVPAnP na função custo. Como resultado, a validação deste algoritmo foi realizada com base no teste ANOVA, com p-valor < 0,01, evidenciando uma melhora significativa na precisão do modelo no conjunto de teste para a rede 1D CONVNET. Os erros médios foram de MSPE = 4,49%, RMPSE = 2,23% e EQM = 0,09, para 600 salas de teste. A melhoria com a TF foi de 6 vezes, em termos da média do EQM com validação cruzada (CV = 10), nos cenários com e sem TF. Esses resultados demonstram a eficácia do método de TF para melhorar a precisão da predição do STI. Conclui-se que o algoritmo MVPAnP poderá ser empregado como uma ferramenta de apoio para a avaliação STI em salas de aula, usando apenas a instrumentação associada às medições do TR.



Palavras-chave: Inteligibilidade da Fala. Índice de Transmissão da Fala. Redes Neurais Convolucionais Profundas. Transferência de Aprendizagem. Acústica. Tempo de Reverberação. Inteligência Artificial Generativa.

## ABSTRACT

Acoustic comfort in classrooms interferes with the teaching-learning dynamics. Therefore, classrooms with acoustic problems cause a socioeconomic burden, compromising the efficiency of education. Speech intelligibility is measured by the Speech Transmission Index (STI). However, measuring this index is complex and expensive. In current literature, there are STI predictive models that use Reverberation Time (RT) as a regression variable. However, these models do not consider the spectral effects of the dual time-frequency location of the Room Impulse Response (RIR) signal, as well as the spectral content of the background noise. This work aimed to generate a predictive model of the STI by applying a deep one-dimensional convolutional neural network (1DCONVNET). However, due to the difficulty in performing *in situ* measurements of the STI, and with the objective of providing robustness to the estimates of the 1DCONVNET network, a new Transfer Learning (TF) algorithm was proposed, called Non-Parametric Projective-Adaptive Variational Minimization (NP-PAVM). NP-PAVM aims to optimize a generative model trained with simulated data and transfer the generalization to a measurement dataset. The NP-PAVM algorithm is based on Generative Artificial Intelligence, via Variational Autoencoders (VAE) networks. The VAE network generated an aggregated posterior distribution in the source latent space that was applied as a parameterization of the distribution of the target set. The relationship between the source and target sets was achieved via Kernel Principal Component Regression (KPCR) on the latent spaces generated by the VAE network. The size of the training and validation sets of the VAE network was 25000 and 5000 classrooms, respectively. The 1DCONVNET network has two input signals, the RIR, simulated via the Image Method and the background noise (BGN), both inputs were processed by Power Spectral Density (PSD), the output was the STI. The experimental validation of the 1DCONVNET network was carried out in 13 classrooms, with 301 measurements of TR - ISO 3288-2 and 90 measurements of STI - IEC 60268-16. From these measurements, a test set with 600 classrooms was generated. This set was used to evaluate the 1DCONVNET network in scenarios with and without TF. This scenario configuration is represented by coupling the NP-PAVM algorithm in the cost function. As a result, the validation of this algorithm was carried out based on the ANOVA test, with p-value < 0.01, showing a significant improvement in model accuracy in the test set for the 1D CONVNET network. The average errors were MSPE = 4.49%, RMPSE = 2.23% and MSE = 0.09, for 600 test classrooms. The improvement with TF was 6 times, in terms of the mean of the MSE with cross-validation (CV = 10), in the scenarios with and without TF. These results demonstrate the effectiveness of the TF method in improving the accuracy of STI prediction. It is concluded that the NP-PAVM algorithm can be used as a support tool for STI assessment in classrooms, using only the instrumentation associated with RT measurements.



Keywords: Speech Intelligibility. Speech Transmission Index. Deep Convolutional Neural Networks. Transfer Learning. Acoustics. Reverberation Time. Generative Artificial Intelligence.

## LISTA DE FIGURAS

## LISTA DE QUADROS E TABELAS

## LISTA DE ABREVIATURAS OU SIGLAS

| | |
|---|---|
| 1D | Espaço unidimensional |
| 1DCONVNET | *One Dimensional Convolucional Neural Network Model* |
| 2D | Espaço bidimensional |
| ABNT | Associação Brasileira de Normas Técnicas |
| ACP | Análise de Componentes Principais |
| AFE | *Automatic Feature Extraction* |
| ANOVA | *Analysis of Variance* |
| ANSI | *American National Standard Institute* |
| ANSI/ASA | *American National Standards Institute / Acoustical Society of America* |
| BB93 | *Building Bulletin 93* |
| CDF | Distribuição acumulada |
| Conv1D | Camada de Rede Neural de Convolução unidimensional |
| CV | *Cross Validation* |
| dB(A) | Escala decibel na ponderação do filtro A |
| DMP | Diferença Minimamente Perceptível |
| EMQP | Erro Médio Quadrático Percentual |
| EQM | Erro Quadrático Médio |
| EQMP | Erro Quadrático Médio Percentual |
| ESTI | Índice de Transmissão de Fala Estendido |
| FDA | Função Distribuição Acumuladas |
| Hz | Frequência |
| IEC | *International Electrotechnical Commission* |
| INR | *Impulse response to Noise Ratio* |
| ISM | *Image Source Method* |
| ISO | *International Organization for Standardization* |
| JND | Just Noticeable Difference |
| KDE | *Kernel Density Estimation* |
| KPCA | *Kernel Principal Component Analysis* |
| KPCR | *Kernel Principal Component Regression* |
| K-S | Teste de Kolmogorov-Smirnov |

| | |
|---|---|
| MAPE | *Mean Absolute Percentage Error* |
| MLP | *Multilayer Perceptron* |
| MMD | *Maximum Mean Discrepancy* |
| MSE | *Mean Squared Error* |
| MSPE | *Mean Squared Percentage Error* |
| MTF | *Modulation Transfer Function* |
| MTI | *Modulation Transmission Index* |
| MVPAnP | Minimização Variacional Projetiva-Adaptativa não Paramétrica |
| PAIR | Perda Auditiva Induzida por Ruído |
| PCA | *Principal Component Analysis* |
| PCR | *Principal Component Regression* |
| PSD(BGN) | Densidade Espectral aplicada no Ruído de Fundo |
| PSD(RIR) | Densidade Espectral aplicada na Resposta Impulsiva da Sala |
| RF | Ruído de Fundo |
| RIR | *Room Impulse Response* |
| RIS | Resposta impulsiva da sala |
| RMSPE | *Root Mean Squared Percentage Error* |
| RSR | Relação Sinal Ruído |
| SLITs | Sistemas lineares e invariantes no tempo |
| SNR | *Signal to Noise Ratio* |
| STI | Speech Transmission Index |
| TF | Transferência de Aprendizagem |
| TR | Tempo de Reverberação |
| VAE | *Variational Autoencoder* |
| wav | *Waveform audio format* |

## LISTA DE SÍMBOLOS

| | |
|---|---|
| % | percentual |
| ® | Marca registrada |
| $\Sigma$ | Somatório de números |
| © | Copyright |
| $(\cdot)^T$ | Operador de transposição matricial |
| $\lim_{x\to\infty} f(x)$ | Operador limite |
| $\frac{\partial^2 p}{\partial x^2}$ | Derivada Parcial |
| $1_N$ | Matriz unitária quadrada com dimensão, $N \times N$ |
| $1_N^T$ | Matriz unitária transposta |
| $\arg\min_{\lambda\in\Lambda} f(\lambda)$ | Mínimo global de f sujeito às restrições $\Lambda$ |
| $\arg\max_{0\le x\le k} f(x)$ | Máximo global de f sujeito às restrições de domínio |
| $\hat{\mathbb{C}}$ | Matriz de variância semiempírica |
| $\mathcal{L}_{RE}$ | Função objetivo do erro de reconstrução |
| $\min_{\Phi}$ | Mínimo de determinada função $\Phi$ |
| $L_{BGN,k}$ | Nível de ruído de fundo (dB) |
| $L_P$ | Nível de Pressão Sonora (dB) |
| $L_{eq,T}$ | Nível de pressão sonora contínuo equivalente (dB) |
| $L_{eq}$ | Nível de pressão sonora equivalente |
| $L_{op,k}$ | Nível operacional da fala (dB) |
| $SNR_{eff}$ | Relação sinal ruído efetiva |
| $\ddot{p}$ | Segunda derivada temporal |
| $p_{ref}$ | Pressão de referência |
| $x_{m,i}$ | Amostra de treinamento não normalizada |
| $x'_{m,i}$ | Amostra de treinamento normalizada |
| $\mathcal{A}_\lambda$ | Configuração ótimo de modelo hiperparametrizado |
| $\mathcal{D}_{KL}$ | Métrica de divergência de Kullback-Leibler |
| $\alpha_i$ | Coeficiente de absorção de som médio da parede $i$ |
| $\alpha_i(\hat{k}\|k)$ | Estimativa das classes no espaço alvo |

| | |
|---|---|
| $\beta_k$ | Autovetor ortogonal |
| $\lambda^*$ | Conjunto ótimo de hiperparâmetros |
| $\rho_0$ | Massa específica |
| $\rho_K$ | Estimativa da densidade kernel para o ponto |
| $\Delta$ | Operador Laplaciano |
| $AL_{cons}$ | *Articulation Loss of Consonants* |
| C50 | Claridade (dB) |
| D50 | Definição |
| dim($\mathcal{F}$) | Operador dimensional |
| G | Intensidade/Força (dB) |
| $\mathcal{H}$ | Espaço de Hilbert |
| H0 | Hipótese nula (H0) |
| $\mathcal{L}$ | Função objetivo/custo |
| $\mathcal{L}(\mathrm{x}$ | Função objetivo global |
| L1pm | Nível de pressão sonora à 1 metro da fonte |
| LAeq | Nível de Pressão Sonora Equivalente ponderado em A |
| LF | Distribuição Espacial Lateral |
| $m^3$ | Metro cúbico |
| U50 | Razão de Som Útil-a-Prejudicial |
| $\mathrm{z}$ | Coordenadas do espaço latente não linear |
| $\Phi(\cdot)$ | Operador que realiza um mapeamento de redução dimensional |
| $\Omega(\Phi)$ | Função regularizadora |
| $DIST(\cdot)$ | Métrica para o distanciamento de distribuições |
| $E(t)$ | Energia integrada recebida durante um intervalo de tempo |
| $F$ | Frequência de modulação |
| $V$ | Matriz de autovetores |
| $VAR(\cdot)$ | Variância das amostras dos domínios $\mathcal{D}$ e |
| $e$ | Número de Euler |
| $ln$ | Logaritmo natural |
| $m(f_m)$ | Fator de redução de modulação |
| $m(F)$ | Fator de redução de modulação na frequência |
| $\mathcal{D}$ | Domínio de origem, ou fonte, é dado por $\mathcal{D} = \{\mathcal{X}, \mathcal{P}(\mathcal{X})\}$ |
| $\mathcal{N}(0,1)$ | Função densidade de probabilidade gaussiana |

| | |
|---|---|
| $\mathcal{P}(\mathcal{X})$ | Distribuição de probabilidade marginal de $\mathcal{X}$ |
| $\mathcal{T}$ | Domínio alvo é dado por $\mathcal{T} = \{\mathcal{Y}, \not{f}\}$ |
| $\mathcal{X}$ | Espaço dos atributos dados por $\{x_{m,i}\}_{i=1}^{j=2}$ |
| $\mathcal{Y}$ | Espaço vetorial correspondente ao classificador |
| $\not{f}$ | Função de distribuição condicional probabilística |
| $\Psi$ | Matriz de mapeamento no espaço dos atributos |
| $\delta$ | Delta de Dirac |
| $\mu(x_{m,i})$ | Média das amostras de treinamento - m |
| $\xi$ | Vetor dos coeficientes de regressão |
| $\pi$ | Número pi |
| $\sigma(x_{m,i})$ | Desvio padrão das amostras de treinamento - m |
| $\mathbf{1}$ | Matriz com todas as entradas iguais a unidade |

**SUMÁRIO**

# 1 INTRODUÇÃO

O ruído como estressor ambiental se apresenta como um dos principais elementos subjacentes ao surgimento/desencadeamento de doenças das mais variadas formas e patologias (GUO et al. 2017). A compreensão dos efeitos dos ruídos depende dos fatores que o caracterizam, tais como a fonte, o meio (canal) de transmissão e o receptor. Cada um desses itens elencados pode-se manifestar das mais variadas formas no cotidiano, mesmo com ou sem a consciência dos indivíduos afetados. Como no caso de ambientes escolares, onde o ruído é duplamente oneroso. Ele interfere na relação ensino aprendizagem e pode causar, nos alunos e professores doenças correlatas à prolongada exposição (BASNER et al., 2014).

Em ambientes escolares, Connolly et al. (2019) esclareçam os principais efeitos nocivos do ruído no desenvolvimento psicossocial das crianças. Neste estudo ainda foi elencado que tais efeitos podem perdurar até as fases posteriores do desenvolvimento da criança, chegando a emergir manifestações na fase adulta. No contexto do sistema auditivo, existem relatos da manifestação de zumbido (*tinnitus*); em casos mais severos, há relatos de Perda Auditiva Induzida por Ruído (PAIR). Já fora do sistema auditivo, há relatos de: perturbações do sono, stress, fadiga mental, diminuição da concentração, problemas de comunicação, irritabilidade, distúrbios cognitivos e de aprendizagem e doenças cardiovasculares (WEN et al., 2019).

Independentemente do nível educacional, para a boa execução das atividades acadêmicas, a qualidade da comunicação entre o professor e alunos é um requisito primordial. Notadamente, o excesso de ruído, ou mesmo, de condições acústicas deficitárias, tendem a contribuir negativamente para a qualidade da fala (DOCKRELL; SHIELD, 2006; KLATTE; BERGSTRÖM; LACHMANN, 2013).

No contexto de acústica de salas, o Tempo de Reverberação (TR) é um dos descritores acústicos consolidados e, por consequência, amplamente utilizado em projetos arquitetônicos. Recentemente, o TR ganhou atenção em sistemas de reconhecimento automático de fala, devido à possibilidade de realizar a convolução do sinal de voz com uma resposta impulsiva e produzir o efeito de auralização (TENENBAUM et al, 2018; TANG; MENG; MANOCHA, 2020).

No entanto, de acordo com Van Schoonhoven, Rhebergen e Dreschler (2017) o tempo de reverberação por si só apresenta várias desvantagens quando se trata de

definir uma interpretação quanto à qualidade da fala, que é representada pelos descritores acústicos da inteligibilidade da fala. Sabe-se que o TR é um descritor psicofisiológico e, como tal, pode ser visualizado por meio de múltiplas bases de significado (HARRIS, 1991). Nesse sentido, o desenvolvimento de novas formas de avaliação diagnóstica da qualidade acústica em salas de aula, que sejam, simultaneamente, rápidas e práticas, são de suma importância. Portanto, este trabalho objetiva contribuir com o estado da arte correspondente por meio do desenvolvimento de ferramentas computacionais.

## 1.1 JUSTIFICATIVA

Segundo a normativa ISO 3382:2 (ISO, 2008), que define os procedimentos da medição do TR em salas ordinárias, somente a medição do TR pode ser uma via ineficaz para a caracterização das salas. Adicionalmente, nenhuma relação direta pode ser construída para determinar os valores do TR ditos ideais sem ponderar variáveis como o volume da sala, o ruído de fundo e seu uso pretendido, ou seja, a finalidade.

Ao contrário, o Índice de Transmissão de Fala, ou Speech *Transmission Index* (STI), normatizado segundo a IEC 60268-16:2011 (IEC, 2011), foi desenvolvido originalmente por Houtgast e Steeneken (1973) objetivando superar as desvantagens elencadas previamente. O STI oferece vantagens teóricas, que incluem a avaliação da deterioração do sinal acústico entre as posições da fonte e do receptor dentro do recinto sob avaliação, considerando o efeito combinado do tempo de reverberação e do ruído de fundo.

O ruído de fundo é representado pelo efeito da relação sinal-ruído. Para este fim, aplica-se o conceito de função de transferência de modulação, ou *Modulation Transfer Function* (MTF), que calcula a deterioração do sinal com base nas relações de profundidade, ou seja, das amplitudes das ondas senoidais moduladas em canais de transmissão.

A medição do STI segundo a IEC 60268-16:2011 (IEC, 2011) exige instrumentação específica, que pode ser difícil para os países em desenvolvimento adquirirem, devido ao seu alto custo. Por outro lado, na literatura corrente, existem diversos modelos que buscam usar o TR como uma variável preditiva para o STI.

Contudo, nesses modelos, apenas algumas bandas de oitavas medidas do TR são usadas para calcular o STI.

Constata-se, portanto, que o presente trabalho pode vir a contribuir significativamente para o campo de conhecimento em tela.

## 1.2 HIPÓTESE

Os modelos preditivos atuais são vastos e desconexos, na medida em que usam diferentes noções de cálculo do TR e não há padronização relativa às bandas de oitava aplicadas como variável de regressão (TANG; YEUNG, 2004; ESCOBAR; MORILLAS, 2015; NOWOŚWIAT; OLECHOWSKA, 2016; LIU et al. 2020). Dessa forma, devido à importância do STI na acústica de salas, reconhece-se a necessidade do desenvolvimento de modelagens robustas para modelos preditivos do STI

Os modelos correntes utilizam o tempo de reverberação em bandas de oitava como variável de regressão. Ocorre que, a modelagem via redes neurais artificiais convolucionais para a aprendizagem profunda propiciam o desenvolvimento de um modelo que usa todo o sinal captado da resposta impulsiva da sala, advindo da medição direta do TR. Partindo disso, e ao otimizar as redes neurais com a heurística de transferência de aprendizagem, este trabalho pode vir a preencher essas lacunas nos modelos, e ao mesmo tempo, contribuir com o estado da arte.

## 1.3 OBJETIVOS

### 1.3.1 Objetivo geral

Este trabalho objetiva desenvolver um novo algoritmo preditivo para o Índice de Transmissão da Fala, STI. Para tanto, aplicam-se os métodos de redes neurais artificiais profundas acopladas com heurísticas de transmissão de aprendizagem profunda. O algoritmo proposto nessa tese, apresenta uma abordagem híbrida entre a transferência de aprendizado baseado em projeção com o aprendizado baseado na métrica de distância de distribuição, por meio da otimização da similaridade de subespaços latentes via projeções. O algoritmo é denominado de transferência de

aprendizagem via Minimização Variacional Projetiva-Adaptativa não Paramétrica (MVPAnP).

### 1.3.2 Objetivos específicos

Para atingir o objetivo geral são elencados os seguintes objetivos específicos que, a serem realizados em ordem sistemática:

a) medir o tempo de reverberação e o índice de transmissão da fala;
b) criar o banco de dados das respostas impulsivas das salas aplicando o método das imagens;
c) adquirir o banco de dados do ruído de fundo ambiental;
d) implementar a topologia de treinamento das redes neurais artificiais de aprendizagem profunda;
e) realizar o mapeamento multidimensional das entradas via filtros convolucionais;
f) pré-processar as entradas do modelo de aprendizagem profunda via transformação da densidade espectral aplicada nas respostas impulsivas e no ruído de fundo;
g) desenvolver o método de transferência de aprendizagem via projeções em espaços latentes, e minimizar a similaridade entre o conjunto de respostas impulsivas simuladas e as respostas impulsivas medidas (ver item – “a”);
h) desenvolver a representação de *autoencoder* aplicando o método de Análise do Componente Principal via Kernel e comparar o método proposto de transferência de aprendizagem com o método da Discrepância Média Máxima;
i) validar e consolidar o método de predição do STI proposto neste trabalho, perante os modelos na literatura e através de medições *in situ* do STI.

## 1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho foi organizado em capítulos. O capítulo 1 apresenta a introdução do trabalho. Nele são mostradas as justificativas e hipóteses norteadoras para o desenvolvimento da tese.

No capítulo 2 é realizada a revisão da literatura, que trata das definições físicas do tempo de reverberação e do índice de transmissão da fala sob a ótica de sinais e sistemas. Contextualiza-se o cenário atual das medições de salas de aula em termos do tempo de reverberação, e o ruído de fundo e STI.

Já no capítulo 3 é desenvolvida a metodologia do trabalho, com definição dos modelos de rede neural convolucional unidimensional. Apresenta-se o algoritmo de treinamento e a respectiva hiperparametrização de treinamento. Descreve-se também a forma de criação e aquisição do conjunto de dados de treinamento e validação. Em seguida, é descrito o método de aprendizagem por transferência.

O capítulo 4 apresenta os resultados e discussões, ao passo que o capítulo 5 traz conclusões e delineou-se as sugestões para trabalhos futuros. Por fim, em anexo, após as referências bibliográficas, são relacionados todos os códigos, desenvolvidos para a implementação do método proposto, em linguagem Python.

# 2 REVISÃO DE LITERATURA

## 2.1 ACÚSTICA DE SALAS

Os fenômenos físicos decorrentes da geração e propagação de sons em ambientes fechados são estudados em diversos contextos, tais como arquitetônicos, de engenharia e físicos. De maneira geral, esse campo de estudo é denominado de acústica de salas. Todavia, a interação do homem em tais ambientes exige modelos que devem incluir os fatores biológicos do som, como os abordados pela fisiologia e psicologia, o que geram a sua decorrente percepção.

Assim sendo, tal fato revela o caráter indissociável da interdependência entre a concepção matemática do modelo de propagação do som e a sensibilidade humana em relação à percepção dele. Logo, guiado pelas prévias assumpções, em seguida demonstram-se as abordagens da fenomenologia dos sons em ambientes fechados, em que se busca encadear as definições aplicadas no contexto de acústica de salas.

### 2.1.1 Modelagem via equação da onda linearizada

Segundo Harris (1991), as relações e modos de propagação do som em ambientes fechados podem ser compreendidas sobre as perspectivas da modelagem via a equação da onda generalizada. Também pode-se utilizar os modelos baseados na teoria difusão da energia em meios contínuos, os quais fazem uso da física estatística para modelar a relação entre a propagação e a interação do som num ambiente fechado, incluindo também o efeito de sua geometria. Por meio da formulação da equação da onda, podem ser retiradas abstrações que permitem avaliar a acústica de salas sob a perspectiva quantitativa, representada pela teoria de sistemas lineares invariantes no tempo e os modos de normas de vibração.

A equação da onda é uma equação diferencial parcial do tipo parabólica, de segunda ordem e linear (EVANS, 1997). Fisicamente, Vorländer (2007) realizou algumas ponderações energéticas sobre a propagação do som via equação da onda, ao acoplar variações infinitesimais de pressão e seu respectivo efeito variacional na quantidade de movimento em um ponto material. Tal efeito caracteriza-se pela alteração da velocidade do ponto no meio contínuo, dada pela Eq. 1,

$$\frac{\partial^2 p}{\partial x^2} - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = -\rho_0 \frac{\partial q}{\partial t} \tag{1}$$

em que $p$ é a pressão, $c^2$ é uma constante relacionada ao processo adiabático, da derivada da pressão em relação a massa específica, $c^2 = dp/d\rho$, em outros termos, c é velocidade do som no meio material, $\rho_0$ é a massa específica do meio de transmissão, $q$ é a quantidade de movimento, $x$ é a posição e $t$ é o tempo. A Eq. (1) pode ser generalizada para o espaço n-dimensional, como,

$$\Delta p - \frac{1}{c^2}\ddot{p} = -\rho_0 \dot{q} \tag{2}$$

em que $\Delta$ é o operador Laplaciano, $\ddot{p}$ representa a segunda derivada temporal da pressão, e $\dot{q}$ representa a primeira derivada temporal da quantidade de movimento. O operador laplaciano aplicado em $p$ é dado por

$$\Delta p = \sum_{i=1}^{n} \frac{\partial^2 p}{\partial x_i^2} \tag{3}$$

sem a perda de generalização, considera-se a pressão como campo vetorial devidamente definido por funções harmônicas ou por expansões harmônicas, dadas por $p : \mathbb{R}^n \to \mathbb{R}$, com $n = 3$, o que corresponde ao espaço tridimensional. Evidentemente, a resolução da Eq. (1) está sujeita às restrições das condições iniciais e de contorno do problema, levando em consideração, quando apropriado, simplificações no modelo, tais como amortecimento, geometria da clausura.

Para a formulação da solução da equação da onda de forma generalizada, adotou-se a abordagem de Vér e Beranek (2005) via mapeamento funcional no domínio da frequência para excitações randômicas no campo acústico em ambiente fechado. Nesse caso, a solução generalizada da Eq. (1), no domínio da frequência pode ser aproximada pela teoria de análise aleatória. Assim, os processos estacionários não determinísticos são caracterizados pela sua função de densidade espectral de potência, em inglês, *Power Spectral Density* (PSD) e pela função de densidade espectral cruzada.

A função de densidade espectral de potência (PSD) de uma variável física, como a pressão sonora $p(t)$, é definida como uma função de magnitude real $S_P(\omega)$ em $(N/m^2)^2/Hz$,

$$S_P(\omega) = \lim_{T\to\infty} \frac{2}{T} \left| \int_0^T e^{-i\omega t} p(t) dt \right|^2 \tag{4}$$

Da mesma forma, a função de densidade espectral de $S_Q(\omega)$ em $(m^3/s)^2/Hz$, para uma fonte de pontual, como volume-velocidade $Q(t)$, é

$$S_Q(\omega) = \lim_{T\to\infty} \frac{2}{T} \left| \int_0^T e^{-i\omega t} Q(t) dt \right|^2 \tag{5}$$

as funções PSD da resposta de pressão sonora e da excitação da fonte sonora são relacionadas pela Função de Resposta em Frequência (FRF) como

$$S_p(x, y, z, \omega) = \sum_n p_n^2(x, y, z, \omega) S_Q(\omega) \tag{6}$$

onde $S_p(x, y, z, \omega)$ $(N/m^2)^2/Hz$, é a resposta de PSD de pressão sonora e $p_n^2(x, y, z, \omega)$ é a função de resposta em frequência modal. Outro resultado útil da teoria de análise aleatória é que, se várias fontes $Q_1(t), Q_2(t), \cdots, Q_a(t)$ são estatisticamente independentes, então a correlação cruzada entre qualquer par de fontes é zero, e a resposta de PSD da pressão sonora total é igual à soma das respostas de PSD devido às fontes individuais, ou seja,

$$S_p(x, y, z, \omega) = \sum_a S_{p_a}(x, y, z, \omega) = \sum_a \sum_n p_{an}^2(x, y, z, \omega) S_{Q_a}(\omega) \tag{7}$$

neste contexto, tem-se exposto os princípios fundamentais da análise aleatória ao investigar as propriedades das funções de densidade espectral (PSD) e suas relações em resposta a fontes sonoras diversas.

### 2.1.2 Abordagem de sinais e sistemas

Ao avaliar o caso particular da Eq. 6, a relação entre pressões associadas a dois pontos, $S_P(\omega)$ e $S_Q(\omega)$, do ambiente enclausurado, expressa a função de transferência entre a fonte e o receptor. A noção de função de transferência é útil, na medida que facilita a manipulação das condições iniciais e pode expressar as variações de pressão como valores relativos a um determinado referencial posicional e temporal, com o qual aplica-se o princípio da superposição.

Dessa forma, a equação da onda, Eq. (1), advém da simplificação da modelagem que visa estabelecer uma correspondência linear. Para Vorländer (2007), os sistemas lineares e invariantes no tempo podem ser caracterizados por meio de sua função de transferência. Todavia, antes de adentrar nesses paradigmas, segundo Vorländer (2007), é usual expressar excitações como rápidas flutuações finitas localizadas no espaço-tempo com conteúdo espectral amplo. Para tanto, a Eq. (8) apresenta a forma genérica desse tipo de excitação, qual seja,

$$q(r) = Q\delta(r - r_0) \tag{8}$$

em que $q(r)$ é a excitação e $\delta$ é o Delta de Dirac. Como aludido previamente, a função $Q$ manifesta-se como uma classe de funções harmônicas, com componentes ortogonais no domínio da frequência. A generalização dessa condição é embutida pela Eq. 9, a saber,

$$s(t) = \int_{-\infty}^{\infty} Q(\omega) e^{-i\omega t} d\omega \tag{9}$$

em que $s(t)$ representa a excitação no domínio do tempo. Por analogia, obtém-se a respectiva pressão no ponto, que é dada pela Eq. 10 como

$$p(r,t) = \int_{-\infty}^{\infty} p_{\omega}(r) e^{-i\omega t} d\omega \tag{10}$$

em que $p(r,t)$ é a pressão na posição $r$ em função do tempo. Neste contexto, as respostas em função das variações das frequências são dadas de acordo com a característica da fonte. Em geral, pode-se modelar os efeitos da fonte como funções de ondas planas, harmônicas e tridimensionais generalizadas.

Ao fazer o caminho inverso e simultaneamente generalizado para os sistemas lineares e invariantes no tempo (SLITs), pode-se apresentar a saída de qualquer planta linear, através de uma operação de convolução entre a entrada e a função de transferência do sistema. A operação de convolução é dada pela Eq. 11,

$$s'(t) = \int_{-\infty}^{\infty} s(\tau) g(t-\tau) d\tau = \int_{-\infty}^{\infty} g(\tau) s(t-\tau) d\tau \tag{11}$$

em que $s'(t)$ é a saída do sistema, $\tau$ é uma variável de integração auxiliar e $g$ é a função resposta ao impulso. A operação realizada na Eq. 11 é representada na forma compacta como, $s'(t) = s(t) * g(t)$, em que $*$ representa o operador de convolução conforme a Eq. 11. Em termos gerais, quando se representa $g$ no domínio da frequência, essa função recebe o nome de função de transferência, ou função resposta em frequência, do sistema.

Cabe salientar que o uso dessa modelagem em condições reais é complexo, e exige métodos que acoplam condições de contorno em geometrias sem restrições de formas. Para tanto, em simulações diagnósticas, adotam-se outros métodos otimizados para trabalhar nas condições mencionadas. Na seção 3.2, mostra-se a modelagem via o Métodos das Imagens. Na próxima seção, apresenta-se o desenvolvimento desta formulação para obter a caracterização da sala em termos da reverberação.

### 2.1.3 Reverberação

Empiricamente, o trabalho inovador de Sabine (SABINE, 1900) desenvolveu o conceito básico do Tempo de Reverberação (TR). Ele definiu o TR como o tempo gasto, após o sinal ser desligado, para que a energia do sinal de excitação decaia em -60 dB para um nível de referência, ou seja, até o nível de ruído de fundo do recinto. Dessa forma, pode-se relacionar o tempo de reverberação com o volume das salas e com as áreas de absorção sonora, conforme dado na Eq. 12 (LEHMANN; JOHANSSON; NORDHOLM, 2007), sendo a Eq. 12 escrita como

$$T(\bar{\alpha}) = \frac{0{,}161V}{\sum_{i=1}^{6} S_i\,\alpha_i} \tag{12}$$

em que $T$ é o tempo de reverberação em segundos, $V$ é o volume da sala em m$^3$, $S_i$ são as áreas das paredes, em m$^2$, correspondentes a $i$-ésima parede, e $\alpha_i$ é o coeficiente de absorção de som médio da parede $i$.

De acordo com a norma ANSI/ASA S12.60 (ANSI/ASA, 2010), o tempo de reverberação adequado encontra-se entre 0,5 e 0,7 segundos para fala em salas de aula, considerando o ruído contínuo de fundo equivalente em torno de 35 dB(A). Na ISO 3382:2 (ISO, 2008), o tempo de reverberação é calculado com base na teoria da energia do campo difuso, usando a equação de Schroeder, que realiza a integração da resposta impulsiva da sala, dada uma posição do receptor para obter a curva de decaimento de energia, conforme a Eq. 13, qual seja,

$$E(t) = \int_{0}^{\infty} p^2(\tau)d\tau \tag{13}$$

em que $E(t)$ é a energia integrada recebida durante um intervalo de tempo na posição do receptor, $p$ a pressão sonora da resposta impulsiva e $\tau$ é a variável de integração auxiliar. Como o tempo de reverberação depende da frequência, uma filtragem de sinal $p(t)$ é feita nas bandas de oitava de 63 Hz, 125 Hz, 500 Hz, 1 kHz, 2 kHz, 4 kHz e 8 kHz.

## 2.2 MÉTRICAS DA INTELIGIBILIDADE DA FALA OBJETIVO

### 2.2.1 O STI

Historicamente, o desenvolvimento de métricas relativas à qualificação da inteligibilidade da fala tem estado associado ao balanceamento de vários fatores. Todavia, uma característica em comum é que se busca estabelecer uma curva padrão da percepção humana subjetiva da fala, ou seja, procura-se determinar um parâmetro psicométrico.

De tal forma, identificam-se na literatura dois principais grupos de descritores, os objetivos e os subjetivos. As medições subjetivas referem-se aos processos aos quais se usam questionários padronizados e ditados. Esses são compostos de palavras foneticamente balanceadas, sendo solicitado ao ouvinte para identificar a palavra ouvida. Em seguida, consolida-se o percentual de acertos e a este valor se gradua a inteligibilidade. Especialmente, nessa classe de descritores subjetivos, a reprodutibilidade dos resultados é função dos indivíduos sob análise, das condições fisiológicas e psicológicas, e de estressores dos mais diversos. Portanto, a sua aplicação e reprodutibilidade reduzem-se a casos mais restritivos (STEENEKEN; HOUTGAST, 2002).

Por outro lado, uma grande quantidade de descritores acústicos objetivos pode ser empregada para determinar a qualidade da fala. Dentre estes, destacam-se o Tempo de Reverberação (TR) (NOWOŚWIA; OLECHOWSKA, 2016), a Definição (D50) (ANSAY; ZANNIN, 2016), a Claridade (C50) (MARSHALL, 1994; PUGLISI et al., 2018), o Tempo de Decaimento Inicial, em inglês, *Early Decay Time* (EDT) (LIU, et al., 2020), a Razão de Som Útil-a-Prejudicial (U50) (YOUNG-JI, 2017) e o Índice de articulação (IA) (STEENEKEN; HOUTGAST, 2002).

Porém, a qualidade da fala por si só tem significados amplos, que variam de acordo com diferentes conjuntos de objetivos, aplicações e disciplinas. Por exemplo, para determinar se uma sala é adequada para atividades de fala ou canto, deve-se usar valores de referência distintos do mesmo descritor. Além disso, cada um deles pode ter vários significados e graduações, dependendo do uso final da sala.

Em vista da quantidade dos descritores e da dificuldade de padronização destes, o STI apresenta-se como um método objetivo, recomendado pela ISO 9921

(ISO, 2003) e sendo padrão de referência internacional para a medição da inteligibilidade no contexto objetivo. Desse modo, têm-se normativas específicas que recomendam valores de referência para o STI. Para salas de aula, citam-se a norma britânica BB93 (DFE, 2015), a norma finlandesa (SFS 5907, 2004) e a norma alemã DIN 18041:2004 (DIN, 2004).

### 2.2.2 O STI indireto

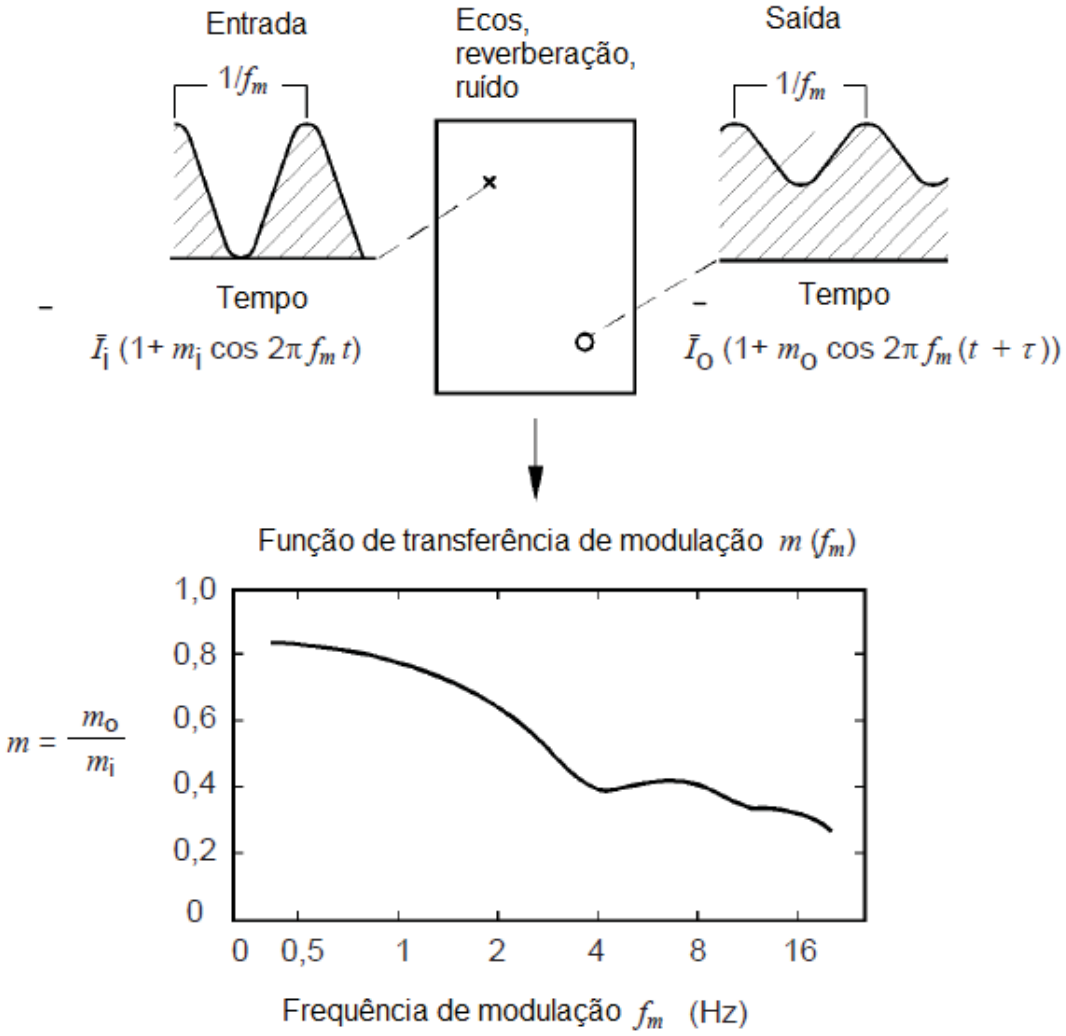
Os procedimentos de medição do STI são estabelecidos na IEC 60268-16 (IEC, 2011). Essa norma não restringe as condições do STI apenas para salas de aula, mas pode ser aplicada de forma genérica nos sistemas de comunicação amplificados artificialmente, como por exemplo: em auditórios, sistemas de alerta de segurança para comunicação em massa e escritórios abertos (LIU et al. 2020; ZHU et al., 2020). Neste trabalho, o STI utilizado foi medido como base no método indireto. Este baseia-se na sua implementação via resposta impulsiva da sala (RIR), com um nível de sinal de excitação em 60 dB(A).

O método do STI indireto (IEC, 2011) considera a degradação entre o sinal modulado gerado na fonte em relação ao sinal recebido na posição do ouvinte, por meio do fator de modulação, $m$, com sua respectiva frequência de modulação, $f_m$. A função de transferência de modulação, $m$, representa a diferença das amplitudes das ondas senoidais moduladas calculadas pela razão entre os sinais de saída e entrada. O efeito da degradação do sinal advém da contribuição do ruído de fundo do ambiente e da reverberação do recinto sob avaliação. Tem, então, que

$$m(F) = \frac{m_0(F)}{m_i(F)} \tag{14}$$

em que $m(F)$ é o fator de redução de modulação na frequência e $F$ é a frequência de modulação. Esse fator é usualmente denominado de função de transferência de modulação, ao passo que $m_i(F)$ e $m_0(F)$ são os índices de modulação entre a amplitude de entrada e a amplitude de saída, respectivamente. A amplitude de saída $m_0(F)$ sofre as interferências do campo reverberante, ecos, distorções não lineares e do ruído de fundo do ambiente. A FIGURA 1 mostra graficamente esse processo.

FIGURA 1 – DERIVAÇÃO DA FUNÇÃO DE TRANSFERÊNCIA DE MODULAÇÃO




FONTE: Adaptado de IEC 60268-16 (IEC, 2011, p. 36).

Para as medições *in situ* do STI indireto a função de transferência de modulação é determinada via a equação de Schroeder (SCHROEDER, 1981). A equação de Schroeder integra a energia da resposta impulsiva da sala, conforme a Eq. 15,

$$m_k(f_m) = \frac{\left|\int_0^\infty h_k^2(t) e^{-j2\pi f_m t}\, d\,t\right|}{\int_0^\infty h_k^2(t)\, d\,t}\left[1 + 10^{-SNR/10}\right]^{-1} \tag{15}$$

em que $m(f_m)$ é fator de redução de modulação, $h(t)$ é a resposta impulsiva da sala, $t$ é o tempo, $f_m$ é a frequência de modulação correspondente às frequências de 0,63,

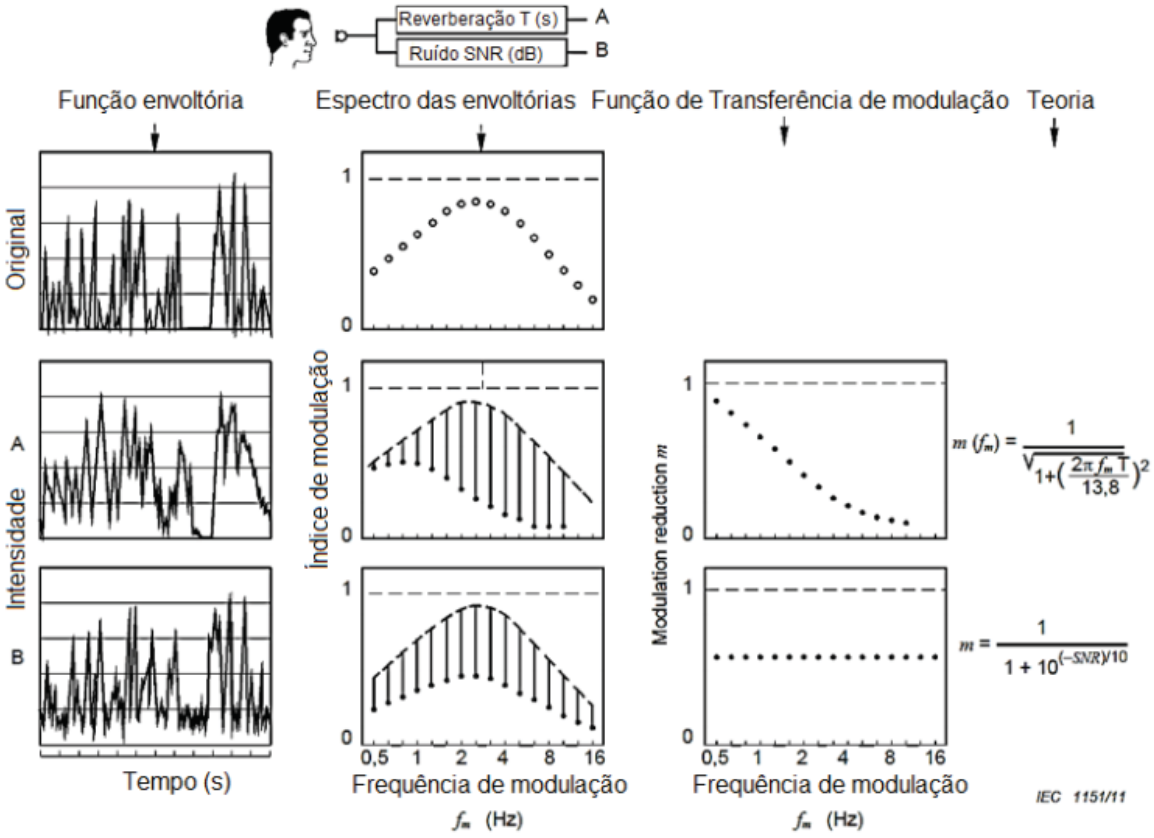
0,80, 1,00, 1,25, 1,60, 2,00, 2,50, 3,15, 4,00, 5,00, 6,30, 8,00, 10,00 e 12,50 Hz, e $SNR$ é a relação sinal-ruído na escala dB. O índice $k$ indica a banda de oitava.

O ruído de fundo relaciona-se com os fatores de modulação, por meio da relação sinal-ruído. O SNR é a diferença entre o nível de fala operacional e o ruído de fundo obtido em suas respectivas bandas de oitava, $k$, de 65 Hz, 125 Hz, 500 Hz, 1 kHz, 2 kHz e 8 kHz, o que é escrito como

$$SNR_k = L_{op,k} - L_{BGN,k} \tag{16}$$

em que $L_{op,k}$ é o nível operacional da fala (dB) e $L_{BGN,k}$ é o nível de ruído de fundo (dB). A IEC 60268-16 (IEC, 2011) recomenda que o $L_{op}$ nível de fala equivalente seja 60 dB(A) a 1 metro de distância em relação à boca do falante. A FIGURA 2 mostra a composição dos efeitos dos canais de reverberação e de ruído sobre o STI.

FIGURA 2 – CONTRIBUIÇÃO DA REVERBERAÇÃO E DO RUÍDO SOBRE O STI




FONTE: Adaptado de IEC 60268-16 (IEC, 2011, p. 36).

Para as medições em campo, usualmente deve-se realizar a equalização do sinal de medição em bandas de oitava até obter o nível operacional com erro de ± 3 dB em cada banda. Ainda assim, no método indireto, o efeito da reverberação e do ruído de fundo são independentes. Todavia, conforme Van Schoonhoven, Rhebergen e Dreschler (2017), durante as medições é usual garantir que a relação sinal ruído seja maior ou igual a +15 dB(A). Com isso, o efeito do segundo termo multiplicativo tende a unidade, mas, para tanto, a relação sinal ruído tende ao infinito. A Eq. 17 indica o efeito separado sobre os fatores de modulação.

$$m_k(f_m)_{SNR\to\infty} = \frac{\left|\int_0^\infty h_k^2(t) e^{-j2\pi f_m t} d\,t\right|}{\int_0^\infty h_k^2(t) d\,t} \tag{17}$$

Adotando-se as definidas propriedades do STI, uma vez determinado os fatores de modulação de $m_k(f_m)$, estes são interpretados por meio da relação sinal-ruído aparente, que é dada por

$$SNR_{eff\,k,f_m} = 10\log_{10}\left(\frac{m(F)}{1-m(F)}\right) \tag{18}$$

em que $SNR_{eff}$ é a relação sinal ruído efetiva, limitada entre de -15 dB até +15 dB. O resultado dessa normalização é denominado de índice de transmissão, expresso pela Eq. 19, em que

$$TI_{k,f_m} = \frac{SNR_{eff\,k,f_m} + 15}{30} \tag{19}$$

Com isso, obtém-se o índice de transferência de modulação, derivado do inglês, *Modulation Transmission Index* (MTI), a saber,

$$MTI_k = \frac{1}{n}\sum_{m=1}^{n} TI_{k,f_m} \tag{20}$$

O valor nominal do STI é

$$STI = \sum_{k=1}^{7} \alpha_k \, MTI_k - \sum_{k=1}^{6} \beta_k \sqrt{MTI_k \, MTI_{k+1}} \tag{21}$$

em que $\alpha_k$ é o fator de ponderação do sexo no espectro de fala e $\beta_k$ é o fator de redundância.

Para a IEC 60268-16 (IEC, 2011), o espectro da voz masculina fornece um valor de STI inferior quando comparado ao espectro de voz feminina, para uma mesma condição experimental. Desta forma, o STI nas medições é usualmente ponderado para o sexo masculino, pois este produzirá uma estimativa conservadora para o STI. Na TABELA 1 mostram-se as classificações subjetivas dos valores do STI

TABELA 1 – GRADAÇÃO SUBJETIVA DO STI

| STI categoria | Ruim - Pobre | Pobre - Regular | Regular - Bom | Bom - Excelente |
|---|---|---|---|---|
| STI nominal | 0,30 | 0,45 | 0,60 | 0,75 |

FONTE: IEC 60268-16 (IEC, 2011)

O STI varia de 0 a 1, em que 0 significa a deterioração total do sinal entre a fonte e o receptor e 1 significa a transmissão perfeita, motivo pelo qual assume-se transmissão de qualidade de 0,75 em diante.

## 2.3 RELAÇÃO ENTRE O RUÍDO, SAÚDE E APRENDIZAGEM

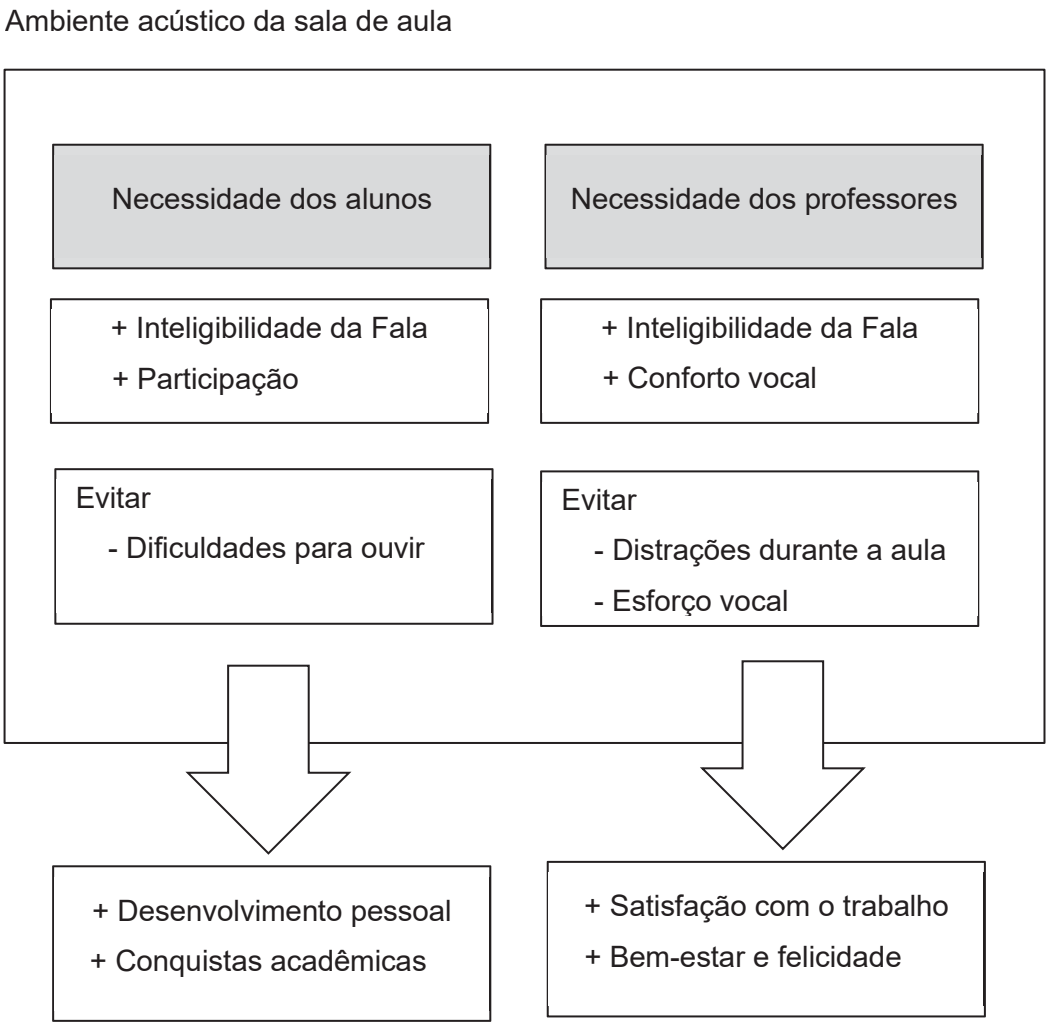
O ruído em contextos escolares para Massonnié et al. (2019), apresenta-se sob duas perspectivas. A primeira em relação à percepção do aluno e a segunda em relação ao professor. Em relação aos professores, destacam-se as decorrências associadas à sua atividade profissional. Portanto, caracteriza-se uma relação no contexto ergonômico, uma vez que se estabelece um vínculo com a qualidade e saúde da voz do professor.

Com base em ambos os contextos, Massonnié et al. (2019) mostram que a relação entre o professor e o aluno se estabelece por meio do ensino e da aprendizagem. Portanto, quando se dá a intersecção das necessidades de

aprendizagem do aluno perante os objetivos de ensino do professor. Tal interseção deve-se dar num ponto ótimo operacional, para assim, estabelecer uma execução de forma coerente e menos danosa possível para o aprendizado do aluno.

Por outro lado, da mesma forma que se compreendeu o contexto em que o professor se encaixava, existe a necessidade de correlacionar as variáveis ditas sociais de crianças, adolescentes e adultos em relação à sua exposição com ruído. Portanto, há uma exacerbada quantidade de variantes que corroboram para o surgimento de doenças em crianças principalmente aquelas na sua primeira infância em relação à exposição prolongada ao ruído (MCGARRIGLE et al, 2019). A FIGURA 3 apresenta os contextos dos alunos e dos professores.

FIGURA 3 – CARACTERIZAÇÃO ACÚSTICA EM SALAS DE AULA




FONTE: Adaptado de Pelegrín-García, Brunskog e Rasmussen (2014, p, 1074).

Para Pelegrín-García, Brunskog e Rasmussen (2014), em salas de aula, a caracterização do ambiente acústico pode ser expressa em termos da perspectiva do controle de ruído e da qualidade acústica. Em termos da qualidade acústica, um dos principais fatores associados na relação ensino aprendizagem é a presença de valores adequados de inteligibilidade da fala e valores adequados para a reverberação.

No que tange, ao controle de ruído, encontra-se a abordagem de buscar soluções mitigadoras, num contexto em que os níveis de ruído se apresentam como elevados. Os ruídos são originários de fontes exógenas, sendo que tais fontes podem variar conforme a disposição no contexto urbano da localização da sala de aula ou da escola. Isso propiciará a entrada de ruído, oriundo de fontes como o tráfego veicular, movimentação de pessoas, ruídos industriais, e outros.

Em relação à qualidade acústica, entendem-se os efeitos associados ao projeto acústico integrado da sala para o desenvolvimento de atividades com o intuito de fala. Assim sendo, existe a preocupação no desenvolvimento de materiais que provocam a alta absorção sonora e simultaneamente propiciem o isolamento do ambiente em relação às fontes externas (PHADKE et al., 2019, WEN et al., 2019).

Lane e Tranel (1971) cunharam, na Psicacústica, o termo Efeito Lambard. Este descreve um fenômeno que foi identificado como, subjetivamente, os indivíduos elevam o seu nível de voz visando sobrepor ruídos externos ao meio da fala. Em outras palavras, no contexto de salas de aula, o professor tende a agravar seus problemas vocais em longo prazo, na medida que utiliza desproporcionalmente a intensidade de sua vocalização com o intuito de melhorar sua compreensão para os alunos. Como consequência, são causados nódulos nas cordas vocais, dores, irritabilidades, insônia e demais manifestações psicofisiológicas decorrentes de tal efeito (WEN et al., 2019).

### 2.3.1 Níveis de exposição de ruído

Rantala e Sala (2015) avaliaram o efeito da acústica de salas sobre a percepção do esforço vocal em professores (n = 40). Os descritores estudados incluíram o TR, o C50, o EDT e o STI, dentre outros. Eles identificaram que os professores relataram incômodo vocal associado primariamente ao ruído durante suas

aulas. Por outro lado, argumentaram que não necessariamente uma condição acústica adequada para a audição, (ou seja, quando se utiliza os descritores de reverberação EDT e TR), gera uma alta correlação com a percepção da fala. Por sua vez, a percepção da fala foi caracterizada apenas pelo STI.

Wen et al. (2019) avaliaram o ruído em 6 escolas de nível secundário, localizadas em Taiyuan, uma cidade chinesa de grande porte. Neste trabalho, foi relatado que as escolas se localizavam em regiões limítrofes a vias de grande fluxo veicular. Com o objetivo de quantificar o efeito do ruído foram realizadas medições internas e externas nas fachadas das salas de aula, durante 20 minutos. Aplicaram-se questionários para compreender a avaliação subjetiva dos alunos em relação ao ruído. O valor máximo Leq foi de 74,2 dB(A), tendo sido verificada a poluição sonora em todas as escolas. Em relação à inteligibilidade, um cenário crítico foi identificado, em que o ruído de fundo mínimo foi de 64,6 dB(A), o que gera uma relação sinal ruído abaixo de -15 dB(A). A principal fonte de incômodo indicado pelos alunos foram fatores originários do tráfego rodoviário, seguido do ruído de salas vizinhas.

Em relação aos efeitos do ruído e à sua respectiva percepção no ambiente escolar, tem-se que, para Shield et al. (2010) e Minichilli et al., (2018), não necessariamente o professor está consciente do impacto do ruído causado em sua saúde. Ao contrário, no âmbito profissional, há relatos de professores que, sim, identificaram a influência negativa do ruído em sua atividade laboral, entretanto, não o reconheceram como risco associado à sua saúde vocal.

Dessa forma, Phadke et al (2019), num estudo transversal por meio da aplicação de questionários (n = 140), objetivaram estimar os fatores que poderiam correlacionar com a saúde vocal dos professores em escolas egípcias. Os resultados mostraram, que mais de 51% dos participantes relataram que aumentavam seu tom de voz. Já 40% dos entrevistados identificaram que o ruído rodoviário interferia significativamente em suas aulas, e enquanto 24% relataram a percepção do ambiente escolar como ruidoso. Os estudos mostraram que houve relação estatisticamente significante entre o ruído e problemas como disfonia e dor no pescoço.

Como foi aludido, o ruído indubitavelmente minora a qualidade da transmissão da fala. Isso incorre em consequências da perda de informação a ser transmitida entre o professor e o aluno, e gera danos nas cordas vocais dos professores. Por este lado,

o efeito da fonte de ruído tem considerável importância. Conforme os trabalhos de Wen et al. (2019) e Phadke et al (2019), as fontes de tráfego rodoviário foram predominantes. Entretanto, o efeito intrínseco, ou seja, o ruído gerado pelos próprios alunos, possui significância.

Conforme o viés apresentado previamente, Levandoski e Zannin (2019) avaliaram o ruído e a sua percepção em professores em duas escolas na cidade de Curitiba, localizada no sul do Brasil. Eles realizaram medições de ruído de fundo e tempo de reverberação, assim como medições externas do ruído ambiental. Os resultados mostraram que os fatores associados aos relatos de percepção do ruído derivam de situações primariamente internas, como o ruído provocado pelos próprios alunos e o ruído transmitido pelas salas próximas.

Ainda conforme Levandoski e Zannin (2019), a percepção de salas ruidosas foi relatada por 55,7% dos professores. Essa porcentagem foi corroborada pelos altos níveis de ruído aferidos na faixa de 74,0 dB(A). Em relação ao tempo de reverberação, obtiveram valores altos, como o de 1,78 s. Por sua vez, conforme Bistafa e Bradley (2000), isso pode estar diretamente relacionado com uma baixa inteligibilidade da fala, portanto, confirmando a percepção identificada pelos professores.

Notou-se uma recorrência na literatura da manifestação de níveis de ruídos fora dos limites estabelecidos por diversas normativas internacionais. Desse modo, constata-se que o ruído é um problema comum nos ambientes acadêmicos. Este não se restringe somente a uma localização geográfica, sendo observado em diversos países.

## 2.4 MEDIÇÕES DE STI EM SALAS DE AULA

Mikulski e Radosz (2011) avaliaram o TR e o STI em 110 salas de aula. As salas foram estratificadas de acordo com o seu tipo arquitetônico e ano de construção. Ao total, as 110 salas foram classificadas em 5 categorias. Em relação ao tempo de reverberação, apenas 3,3% das salas de aula atingiram valores mínimos recomendáveis (TR ≥ 0,65 s). Similarmente, ao avaliarem o STI, cerca de 4,5% das salas apresentaram o valor do STI satisfatório, que foi estipulado em STI ≥ 0,70. Como conclusão, eles afirmaram a necessidade imediata de readaptar as salas de aula avaliadas, diante do notório efeito oneroso na educação dos alunos.

Rabelo et. al (2014) analisaram os descritores tempo de reverberação (TR), nível de ruído equivalente (Leq), e o Índice de Transmissão da Fala (STI) em 18 salas de aula, de 9 escolas públicas do estado de Minas Gerais, no Brasil. Por meio de questionários subjetivos, eles avaliaram a inteligibilidade da fala e compararam estes resultados com os descritores objetivos, TR, Leq e STI. Os resultados indicaram que as salas não atingiram os requisitos mínimos para a inteligibilidade, com um valor mediano de 0,65 para o STI. Em relação ao TR, este variou entre 0,69 s até 2,09 s, considerando as médias das bandas de oitavas de 500 Hz, 1000 Hz e 2000 Hz. O ruído de fundo variou entre 54 dB(A) e 74 dB(A). Eles concluíram que as contribuições de elevados níveis de ruído de fundo e altos tempos de reverberação foram onerosos à qualidade da fala nas salas avaliadas, tomando como referência o valor do teste subjetivo.

Mealings et. al. (2015), por meio de um estudo multidisciplinar, avaliaram o STI, o tempo de reverberação, a relação sinal-ruído e o ruído de fundo em 4 salas de aula com mais de 365 alunos. Os resultados mostraram condições acústicas incongruentes para os ambientes de ensino. O tempo de reverberação médio nas salas consideradas foi de 0,55 s, e nenhuma sala atingiu os requisitos mínimos. Ainda que o tempo de reverberação tenha sido relativamente pequeno, a contribuição do ruído de fundo foi significativa para diminuir a qualidade acústica. Quando as salas eram ocupadas pelos alunos, os níveis de ruído de fundo atingiram até 70 dB(A). Como consequência, o esforço vocal dos professores foi demasiadamente alto. De maneira geral, os professores tiveram que elevar o tom de suas vozes em no mínimo 20,9 dB(A), para obter um nível da relação sinal ruído de, no mínimo, +15 dB(A). Em termos práticos, estes valores de elevação do nível da voz foram relativizados como se os professores tivessem que “gritar” para serem compreendidos. Como consequência dos prévios qualificadores, o STI foi insatisfatório.

Conforme apresentado na seção 2.3, o ruído de fundo degrada o STI. As salas de aula, particularmente aquelas localizadas em países de zonas tropicais, ou em climas com temperaturas mais elevadas tendem a apresentar sistemas de refrigeração, como ar-condicionado, ventiladores e umidificadores. Mecanicamente, estes aparelhos introduzem um ruído adicional, e este ruído adicional compete com a voz do professor e com as conversas entre os alunos.

Neste sentido, Longoni et al. (2016) estudaram o efeito de tais sistemas de refrigeração sobre o STI. Tal efeito foi avaliado com o tipo de controle de ruído on/off, ou seja, quando se desliga a fonte de ruído e faz-se as medições, e, em seguida, liga-se a fonte e procede-se com as medições de STI. Os resultados mostraram um efeito estaticamente significante nas variações das médias do STI.

Por continuidade, no STI indireto, o ruído de fundo é quantificado pelos efeitos independentes entre a reverberação e a relação sinal ruído. Para Choi (2020), as medições do ruído de fundo nem sempre refletirão a condição real da sala de aula. Dessa forma, ele propôs avaliar as salas de ambientes universitários concomitantemente com as suas atividades de ensino e aprendizagem. Para tanto, medições foram feitas em salas de aula ocupadas e desocupadas. Em seguida, mediram-se os seguintes descritores: tempo de reverberação, nível de pressão sonora da fala do professor, ruído de fundo, STI e Razão de Som Útil-a-Prejudicial (U50). Os níveis da fala obtidos foram de 51,5 ± 2,7 dB(A). O tempo de reverberação para as salas ocupadas foi menor do que as salas vazias. O maior STI obtido foi de 0,61 ± 0,01. Com isso, Choi (2020) conclui que medições do STI que emulam as condições reais de aula podem sofrer significativas alterações em relação às salas desocupadas. Assim, infere-se que o ruído de fundo do ambiente é um fator primordial (VAN DE POLL et. al., 2014).

Observou-se, conforme os trechos anteriores, que o STI possui sensibilidade às condições nas salas de aula. Tais efeitos são a composição espectral do ruído de fundo, o tempo de reverberação do ambiente, e de efeitos exógenos, como a introdução de sistemas de refrigeração. Portanto, tais fatores podem ser observados simultaneamente em medições, o que, por consequência, torna a predição do STI mais volátil.

## 2.5 MÉTODOS ALTERNATIVOS DE PREDIÇÃO DO STI

Verificou-se, na seção 2.3, que recorrentemente o STI em geral não possui valores altos. Tais valores derivam de algumas limitações e são difíceis de contornar, devido, sobretudo, às características construtivas do ambiente e a dificuldade intrínseca do processo de medição.

### 2.5.1 Restrições nas medições e modelagens do STI

Conforme a IEC 60268-16 (IEC, 2011), há fontes de erros de medição classicamente identificadas. Algumas destas fontes incluem os fenômenos acústicos, como a diretividade da fonte, a equalização do sinal, o mascaramento, a ressonância harmônica associada aos modos de vibrar da sala e a não estacionariedade do ruído fundo. Esses são alguns dos exemplos de fontes de erro na medição do STI.

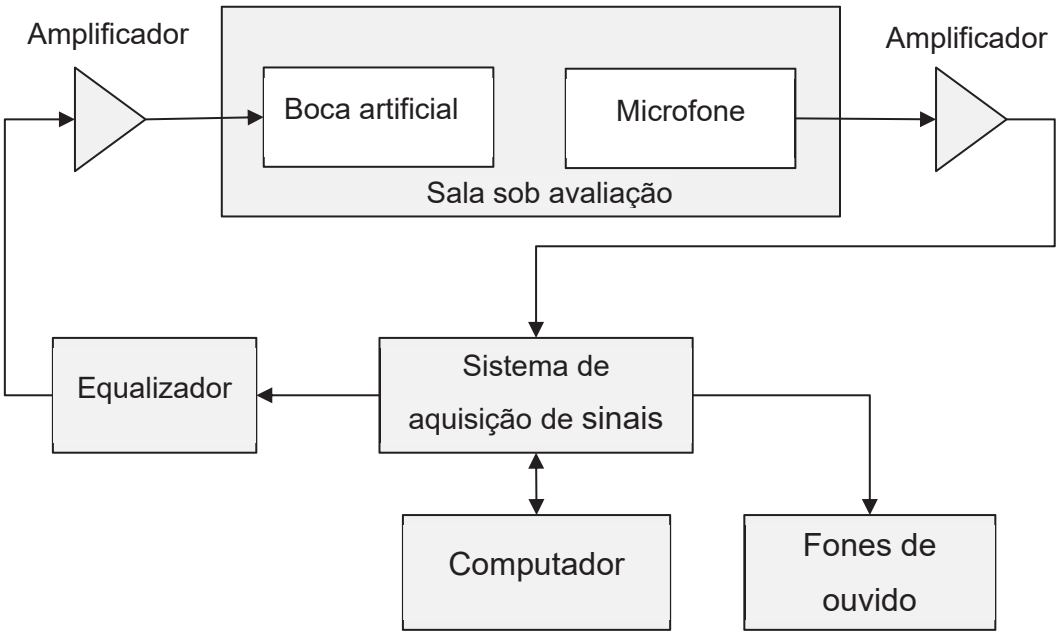
Neste âmbito, Bradley, Reich e Norcross (1999), utilizando o conceito de Diferença Minimamente Perceptível (DMP) mais, conhecido em inglês como *Just Noticeable Difference* (JND), relativizam um valor de erro aceitável para o STI. Este conceito, no contexto do STI, implica a mínima variação perceptível subjetiva do ouvinte. Dessa forma, os erros de medição devem ser menores que a JND, o que, no caso do STI é de 0,03. Assim, ao aplicar a teoria de propagação dos erros, o valor observável do STI deve ter erro de medição até 0,03. Todavia, diversos trabalhos delineiam novas fontes de erros que o STI não abrange. A seguir demonstram-se alguns aspectos relativos dessas limitações.

A primeira limitação refere-se ao uso de sistemas de voz amplificados artificialmente. Esse tipo de aplicação é usual em auditórios, teatros, salas de conferência e espaços grandes em geral. Nestes espaços, o campo acústico não se comporta como um campo difuso. Nos campos acústicos não difusos, surgem efeitos não lineares e, usualmente, sinais de decaimento harmônicos, ou varreduras senoidais, são utilizados como excitação. Todavia, devido à não linearidade e às sobreposições harmônicas essas causam distorções e ressonâncias localizadas no ambiente.

Adicionalmente às fontes de erros do ambiente, as medições do STI necessitam de uma série de equipamentos, conforme mostra a FIGURA 4. Cada

equipamento é calibrado e ajustado para garantir que ruídos elétricos e acústicos não interfiram no STI.

FIGURA 4 – CARACTERIZAÇÃO DA CADEIA DE MEDIÇÃO DO STI




FONTE: Adaptado de Longoni et al (2018, p. 36).

Liu et al. (2020) mostrou que em recintos com volumes entre 97000 $m^3$ e 246000 $m^3$ o STI é estatisticamente diferente das salas ordinárias. Similarmente, Hulva et al. (2017) relatou essa diferença por meio de mapeamento do STI. A hipótese de Hulva et al. (2017) seria verificar qual o fator limite para inferir se uma sala necessitaria de um sistema de amplificação da voz.

Além disso, conforme Liu et al. (2020) aludiram que comumente o STI medido via IEC 60268-16 (IEC, 2011), e o SII, aferido via questionários subjetivos, divergem além do limite do JND de 0,03. Sendo assim, os autores propuseram novas curvas padronizadas entre o STI e os métodos subjetivos. Com tal modificação, pode-se inferir que o STI não necessariamente está alinhado com o teste subjetivo da inteligibilidade para salas do tipo auditório.

A segunda limitação é a medição do STI direto com o ruído de fundo flutuante, ou seja, ruído de fundo não estacionário. Essa limitação no STI deve-se ao fato de não poder garantir a repetibilidade dos resultados, uma vez que a excitação do sinal

senoidal que foi modulada na frequência de modulação, apresentará alto desvio padrão em relação às médias das medições.

De tal modo, Van Schoonhoven, Rhebergen e Dreschler (2017) mostraram que a função de transferência de modulação, nas medições do STI indireto, pode ser obtida satisfatoriamente em ruído não estacionário. Para tanto, eles indicaram o requisito mínimo de 25 dB para a relação impulso-ruído. Essa condição se restringe apenas em excitações do tipo de varredura senoidal. Ainda assim, no mesmo estudo, uma relação entre a relação sinal ruído e relação impulso-ruído foi estabelecida. Com isso, em ruído flutuante, SNR de +15 dB, INR de +25 dB e Leq de +75 dB foram os requisitos mínimos globais estabelecidos.

De toda forma, ao aplicar os limites estabelecidos previamente em salas altamente reverberantes e simultaneamente com a presença de ruídos não estacionários, a medição do STI pode gerar resultados que não reflitam os resultados subjetivos perante os quais o STI foi validado (VAN SCHOONHOVEN; RHEBERGEN; DRESCHLER, 2017). Neste aspecto subjetivo, Van Schoonhoven, Rhebergen e Dreschler (2019) propuseram um modelo denominado de Índice de Transmissão de Fala Estendido (ESTI), que avalia o mascaramento do sinal de excitação em campos reverberantes e com ruído de fundo não estacionário. Uma consequência do fenômeno do mascaramento é a sobreposição silábica e a consequente não distinção dos sons nos indivíduos sob análise. Esse efeito de mascaramento está previsto na IEC 60268-16 (IEC, 2011), e recebe um fator de correção no cálculo do STI. Todavia, em ambientes com ruído não estacionário, esse efeito não recebe nenhum fator de correção.

Ao findar essa seção, ressalta-se que as medições do STI estão susceptíveis a erros sistemáticos. Tais erros são provenientes de várias fontes, como de ruídos elétricos e da manifestação de fenômenos do campo acústico nas salas sob avaliação. Adicionalmente, mesmo com o estabelecimento de condições operacionais de medição dos sinais padronizados, os valores medidos do STI podem destoar da medição subjetiva por meio de questionários (LIU et al., 2020). Portanto, há ainda a necessidade do contínuo desenvolvimento de novos modelos de predição da inteligibilidade da fala.

### 2.5.2 Modelos em desenvolvimento do STI

Muitas aplicações modernas do STI baseiam-se em medidas que visam circundar as limitações explicitadas na seção 2.5.1. Os modelos de predição tradicionais aplicam técnicas de regressão para obter uma estimativa do STI com base na regressão de outros descritores acústicos (TANG; YEUNG, 2004; ESCOBAR; MORILLAS, 2015; NOWOŚWIAT; OLECHOWSKA, 2016; LECCESE, ROCCA, SALVADORI, 2018; LIU et al. 2020).

Como consequência, ao aplicar a abordagem tradicional de regressão, foram destacadas algumas das principais correntes limitações. Com base nas limitações elencadas na QUADRO 1, diversos modelos paramétricos ou não paramétricos, como redes neurais artificiais, emergiram com o objetivo de circundar algumas dessas limitações. Muitos destes métodos aplicaram abordagens não paramétricas para a confecção dos modelos.

QUADRO 1 – CONSOLIDAÇÃO DAS LIMITAÇÕES DOS MODELOS DE PREDIÇÃO DO STI

| CARÁTER DA LIMITAÇÃO | DESCRIÇÃO |
|---|---|
| No tamanho das amostras | Pouca representatividade estatística amostral na obtenção dos modelos de regressão; |
| Na arbitrariedade dos parâmetros de regressão | Adoção arbitrária de bandas de oitava ou uso de médias do nível de ruído (Leq) como variáveis de regressão; |
| Qualidade do ajuste baixa | Modelos de avaliação com baixo coeficiente de correlação de Pearson; |
| Interpretação física | Limitação referente à interpretação física dos modelos, devido à escala dos descritores; |

FONTE: O autor (2023).

Unoki et al. (2017) desenvolveram uma metodologia híbrida baseada em técnicas de otimização numérica e na modelagem do STI. Eles propuseram modelar uma expressão genérica para o decaimento energético de uma excitação dentro de um recinto. Em seguida, obtiveram uma nova expressão para a Função de Transferência de Modulação, que ponderava os efeitos do campo reverberante e do ruído de fundo não estacionário. A conclusão deles foi que o algoritmo pode prever o STI, tanto em ambientes simulados como em medições *in situ*, quanto em ambientes que emulam condições reais dos recintos.

Seetharaman et al. (2018) desenvolverem modelos de redes neurais profundas. As redes convolucionais bidimensionais foram aplicadas como técnicas para o processamento das entradas. A construção dos dados de treinamento baseou-se em bancos de dados simulados das respostas impulsivas e com dados de vozes nítidos que sofreram auralização com as respostas impulsivas das salas. Os áudios foram representados em espectrogramas. A validação dos resultados foi realizada por meio do cálculo do STI indireto. Os resultados determinados apresentaram um erro de 4% da estimativa do STI.

Prodi e Visentin (2019) indicaram algumas das restrições que foram previamente apresentadas na seção 2.5.1, e salientaram que o conteúdo espectral do transiente da voz humana pode ser um fator adicional na complexidade da predição do STI. A hipótese norteadora do novo modelo de predição foi a similaridade de estatísticas entre o método objetivo de predição do STI e o descritor psicoacústico da curva de respostas dos questionários auditivos. A proposta deles foi na avaliação segmentada sobre o truncamento do sinal de excitação e o seu respectivo efeito sobre o mascaramento auditivo. Os autores afirmaram que, com o método de truncamento, objetivaram resultados similares aos descritores subjetivos. Assim sendo, evidenciaram a efetividade do método para condições de medições em campos reverberantes e com ruído não estacionário.

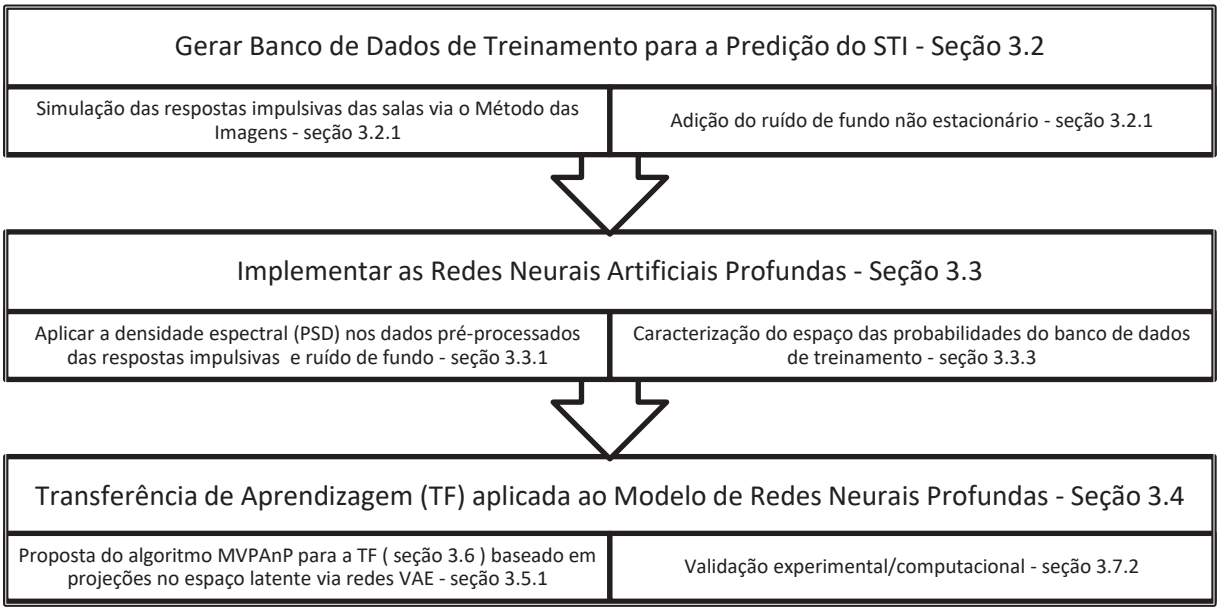
# 3 MATERIAL E MÉTODOS

A predição do STI depende de diversos fatores que rotineiramente são avaliados de forma combinada. Dessa forma, verificou-se, durante a revisão da literatura, uma lacuna nos modelos atuais, tendo em vista que a escolha arbitrária das variáveis de regressão pode gerar modelos que são de difícil reprodutibilidade devido à pouca quantidade amostral. Diante disso, essa seção tem como objetivo descrever a metodologia e os respectivos métodos aplicados para a consolidação de uma nova abordagem preditora para o STI, aplicando as redes neurais artificiais profundas.

## 3.1 ESQUEMA METODOLÓGICO

A metodologia preditiva adotada baseou-se no encadeamento de técnicas numérico-analíticas, alicerçadas nas redes neurais artificiais. De forma complementar, a escolha da modelagem fundamentou-se na perspectiva da possibilidade da adição da interpretação física dos fenômenos acústicos. Com isso, objetivou-se a possibilidade da integração de medições experimentais ao modelo. A FIGURA 5 mostra a metodologia proposta neste trabalho.

FIGURA 5 – ESQUEMA METODOLÓGICO PROPOSTO




FONTE: O autor (2023).

O uso de redes neurais artificiais como modelos não paramétricos para aproximação funcional possui algumas vantagens (GUPTA, 2020). Além disso, a transferência de conhecimento é uma característica que melhora a predição do STI nos cenários de ruído não estacionário e ambientes altamente reverberantes. Neste momento, evidenciam-se a possibilidade da pós-integração de novos dados de treinamento e da interpretação física que advém do uso das redes neurais convolucionais unidimensionais.

Para a implementação das redes neurais artificiais, existe a necessidade da criação de um banco de dados que forneça os pares de entrada e saída do sistema físico ao qual se modela. O banco de dados tem papel crucial na qualidade do modelo, pois quaisquer erros na elaboração deste constituirão em erros de tendência sistemáticos que serão propagados durante a fase de generalização. Uma consequência desse tipo de erro é a baixa qualidade da generalização do modelo para dados que não pertencem ao espaço de treinamento.

## 3.2 MODELO PREDITIVO NORMATIZADO DO STI

No anexo J da IEC 60268-16 (IEC, 2011) tem-se um modelo analítico-preditivo para o STI. Ao aplicar a equação de Schroeder, este modelo determina os fatores de redução de modulação. Neste, os efeitos do ruído de fundo e do tempo de reverberação são independentes, em similaridade ao método de medição indireto. Todavia, a diferença entre estes métodos reside na definição da reverberação.

No método preditivo da IEC 60268-16 (IEC, 2011), avalia-se o tempo de reverberação, ou seja, adota-se um decaimento de -60 dB, a partir do nível de pico do sinal de excitação de uma fonte omnidirecional. Na ausência de medições do tempo de reverberação no recinto, a IEC 60268-16 (IEC, 2011) provê um método de aproximação para o TR.

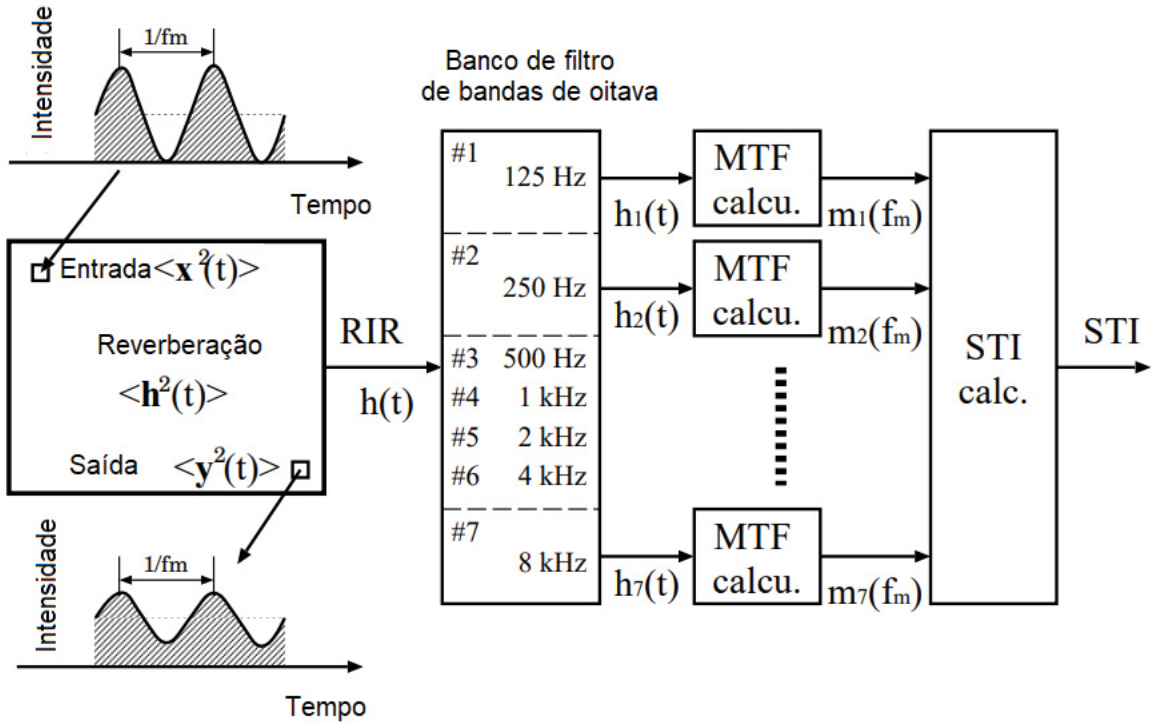
Já, na medição indireta, o efeito da reverberação é dado pela excitação com um nível operacional que emula o espectro da voz masculina, e seu nível é padronizado para 60 dB(A). Uma vez que se emula o espectro da voz masculina, o fator de direcionalidade da fonte deve ser considerado como não direcional. De posse das respostas impulsivas e do ruído de fundo obtém-se os fatores de redução de modulação $m$ via Eq. 22, a saber,

$$m(f_m) \cong \left\{1 + \left(2\pi \frac{T}{13.8}\right)^2\right\}^{-1} \left[1 + 10^{-SNR/10}\right]^{-1} \quad (22)$$

em que $m(f_m)$ são os fatores de redução de modulação, $T$ é o tempo de reverberação para cada banda de oitava de interesse, e $SNR$ é a relação sinal ruído. A obtenção do STI, uma vez determinados os fatores de redução de modulação, foi descrita na seção 2.2.2.

Portanto, o modelo de predição do STI baseia-se na energia da reverberação somada ao efeito do ruído de fundo, em similaridade ao método indireto (ver seção 2.2). As distorções resultantes na amplitude do sinal senoidal em suas respectivas frequências de modulação realizam a caracterização do STI em espectro amplo, conforme mostrado esquematicamente na FIGURA 6.

FIGURA 6 – CÁLCULO TEÓRICO DO STI VIA FUNÇÃO DE TRANSFERÊNCIA DE MODULAÇÃO




FONTE: Adaptado de Unoki et al. (2017, p. 1431).

A IEC 60268-16 (IEC, 2011) apresenta apenas os níveis de referências dos espectros das vozes humanas normalizados para um nível operacional de 60 dB(A).

Neste trabalho, o $L_{op,k}$ foi baseado na norma ISO 9921:2003 (ISO, 2003). A TABELA 2 mostra o nível operacional de fala para a voz humana adotado no modelo preditivo normalizado.

TABELA 2 – NÍVEL OPERACIONAL DA FALA PARA O RECEPTOR A UM METRO DA FONTE

| **Normas** | 125 Hz | 250 Hz | 500 Hz | 1 kHz | 2 kHz | 4 kHz | 8 kHz | Leq |
|---|---|---|---|---|---|---|---|---|
| ANSI | # | 57,2 | 59,8 | 53,5 | 48,8 | 43,8 | 38,6 | 59,5 |
| IEC 60268-16 | 2,9 | 2,9 | -0,8 | -6,8 | -12,8 | -18,8 | -18,8 | 0,0 |

FONTE: ANSI 3.4 (2007).

A restrição desse modelo reside na dificuldade de estimar a direcionalidade da fonte. Essa estimativa visa compensar os efeitos de espalhamento quando a distância entre a fonte e o receptor são relativamente grandes, e a energia direta não é a parcela principal que o receptor recebe (BISTAFA, BRADLEY, 2000)

Neste quesito, Bistafa e Bradley (2000) propuseram o uso de um modelo que mitiga tal restrição, ao argumentar que, a depender da distância na direção de radiação da fonte sonora, o efeito do fator de direcionalidade pode ser desprezível. Em salas de aula com média de 300 m$^3$, a distância limite é dada como a relação da densidade de energia sonora direta para refletida. Isso é compreendido como o limite em que a energia direta é desprezível, e a parcela da energia das reflexões do campo reverberante é a mais significativa. Como consequência, o fator de direcionalidade da fonte pode ser simplificado nos modelos preditivos.

Nesta seção, apresentou-se o procedimento padrão para calcular o STI utilizando o tempo de reverberação. Adicionalmente, ponderou-se o efeito da relação sinal-ruído, ao aplicar do espectro da voz humana conforme as normativas ISO 9921 (ISO, 2003).

### 3.2.1 Criação de conjunto de dados de treinamento

As medições diretas e indiretas do STI são demoradas. Por exemplo, no STI direto, são necessários até 15 minutos para cada ponto de medição. Assim, em salas de tamanho médio, cerca de 300 m$^3$, uma malha uniforme de pontos de medição demorará várias horas. O mesmo acontece para o método indireto. Além disso, a IEC 60268-16 (IEC, 2011) requer um ruído de fundo estacionário durante a medição.

Naturalmente, o ruído de fundo numa sala possui pouca variabilidade, limitando por decorrência o número de amostras de treinamento durante a aquisição.

A solução proposta para estes problemas relacionados à aquisição de sinais advindos de medições experimentais foi a criação de um banco de dados sintético/simulado. Esse banco de dados foi composto pelas respostas impulsivas simuladas das salas via o Método das Imagens (ALLEN; BERKLEY, 1979) e a introdução do ruído de fundo por meio de arquivos de áudio provenientes do banco de dados de Ko et al. (2017).

Em posse do banco de dados, composto pelos sinais acústicos das respostas impulsivas (RIR) e pelos sinais de ruído de fundo ambiental que representaram o ruído de fundo (BGN), aplicou-se a metodologia de predição analítica do STI, conforme exposto na seção 3.2, Eq. 22, para determinar o respectivo valor estimado do STI com base nos dados sintéticos gerados via Método das Imagens.

O Método das Imagens, mais conhecido em inglês como*, Image Source Method* (ISM), foi empregado para obter os sinais simulados no domínio do tempo para as respostas impulsivas (LEHMANN; JOHANSSON; NORDHOLM, 2007). O ISM considera a fonte num recinto, dada pela Eq. 23, qual seja,

$$P(\omega, X, X') = \frac{e^{i\omega(R/c-t)}}{4\pi R} \tag{23}$$

em que $P$ é pressão sonora, $X$ e $X'$ representam a posição da fonte e do receptor respectivamente, e $R$ é o vetor distância entre $X$ e $X'$. As variáveis $X$ e $X'$ pertencem ao espaço euclidiano tridimensional, $X = (x, y, z)$. Após realizar a ponderação das condições de contorno, como exemplo, a adoção da hipótese de paredes rígidas, a resposta impulsiva é dada por

$$P(t, X, X') = \sum_{p=1}^{8} \sum_{r=-\infty}^{\infty} \frac{\delta\left[t - \left(\left|R_p + R_T\right|\right)/c\right]}{4\pi\left|R_p + R_T\right|} \tag{24}$$

em que $P$ é pressão advinda da resposta impulsiva simulada, $\delta$ é o delta de Dirac, e $R_p$ e $R_T$ são as funções de reflexão especular. Para mais detalhes ver Allen e Berkley (1979). Em seguida, calculou-se o tempo de reverberação, T60, para as bandas de

oitava, derivadas do sinal $P(t, X, X')$. A implementação deste foi baseada no programa feito na linguagem FORTRAN de Allen e Berkley (1979). Esse programa foi convertido para a linguagem C++ e emulado na linguagem Python. No total, 30000 salas retangulares virtuais foram projetadas com dimensões aleatórias como o domínio fechado do ambiente, com sistema de coordenadas (x, y, z). O volume mínimo foi 120 $m^3$ e o máximo foi 1000 $m^3$.

A fonte sonora foi alocada no centro da sala retangular, a 1,2 m acima do nível da base, ou seja, com as coordenadas em ($x_s$, $y_s$, 1,2 m), onde $x_s$ e $y_s$ são os pontos médios em relação às dimensões da sala x e y, esses representam também o posicionamento da fonte sonora omnidirecional. O tempo de reverberação, T60, variou na faixa de 0,30 a 2,0 segundos. Os coeficientes de absorção sonora do som correspondentes foram obtidos resolvendo a Eq. 12 para o coeficiente de absorção sonora.

Como explicitado previamente, as amostras dos sinais que emularam o ruído de fundo foram derivadas de um banco de dados de ruídos ambientais (KO et al., 2017). A frequência de amostragem nesses arquivos foi ajustada para 16 kHz. No total, 608 arquivos de áudio no formato *waveform audio format* (wav) foram utilizados. Para evitar a saturação do sinal nos níveis de ruído, a pressão foi normalizada entre zero e 1 Pascal, em cada amostra de áudio. Em seguida, calculou-se a relação sinal ruído, com o auxílio de um banco de filtro passa banda de oitava.

Uma vez obtido o banco de dados no domínio do tempo, para as respostas impulsivas das salas e para o ruído de fundo, foi feito o pós-processamento desses dados utilizando a densidade espectral, dado por,

$$S_{xx}(\omega) = \int_{-\infty}^{\infty} |h(t)|^2 dt = \int_{-\infty}^{\infty} |H(\omega)|^2 \, d\omega \tag{25}$$

em que $h(t)$ é um sinal temporal, e $H(\omega)$ é a transformada de Fourier de $h(t)$. No contexto deste presente trabalho, $h(t)$ representa a resposta impulsiva da sala. O módulo $|H(\omega)|$ é dado por $H^*(\omega)H(\omega)$, em que $H^*(\omega)$ é o complexo conjugado de $H(\omega)$. Estimou-se a densidade espectral de potência via o método das médias dos periodogramas de Welch (WELCH, 1967), sem sobreposição e com aplicação da

função de janelamento de Hann, que é equivalente ao processo de Bartlett (BARTLETT, 1950). Dessa forma, adotou-se uma discretização na frequência com incremento, $\Delta\omega$ de 1 rad/s, com as frequências variando de 0 Hz a 8 kHz.

Utilizou-se esse processamento para uniformizar a dimensão do par entrada-saída das redes neurais artificiais e padronizar, por comprimento de banda, a energia do sinal. Como benefício de aplicar, $S_{xx}(\omega)$ sobre o sinal temporal, pode-se estabelecer uma *proxy* com a formulação do STI indireto via Equação de Schroeder (SCHROEDER, 1981), uma que vez, a função de transferência de modulação pode ser interpretada como a transformada de Fourier complexa da potência da resposta impulsiva para um sinal de energia finita (HOUTGAST; STEENEKEN; PLOMP, 1980, BISTAFA; BRADLEY, 2000). Portanto, ao realizar este processamento se mantém a fidedignidade do conteúdo espectral das respostas impulsivas, ou seja, em vista do Teorema de Parseval, observa-se a invariância da energia do sinal independente da representação do domínio do tempo ou da frequência.

## 3.3 APRENDIZADO PROFUNDO VIA REDES NEURAIS ARTIFICIAIS

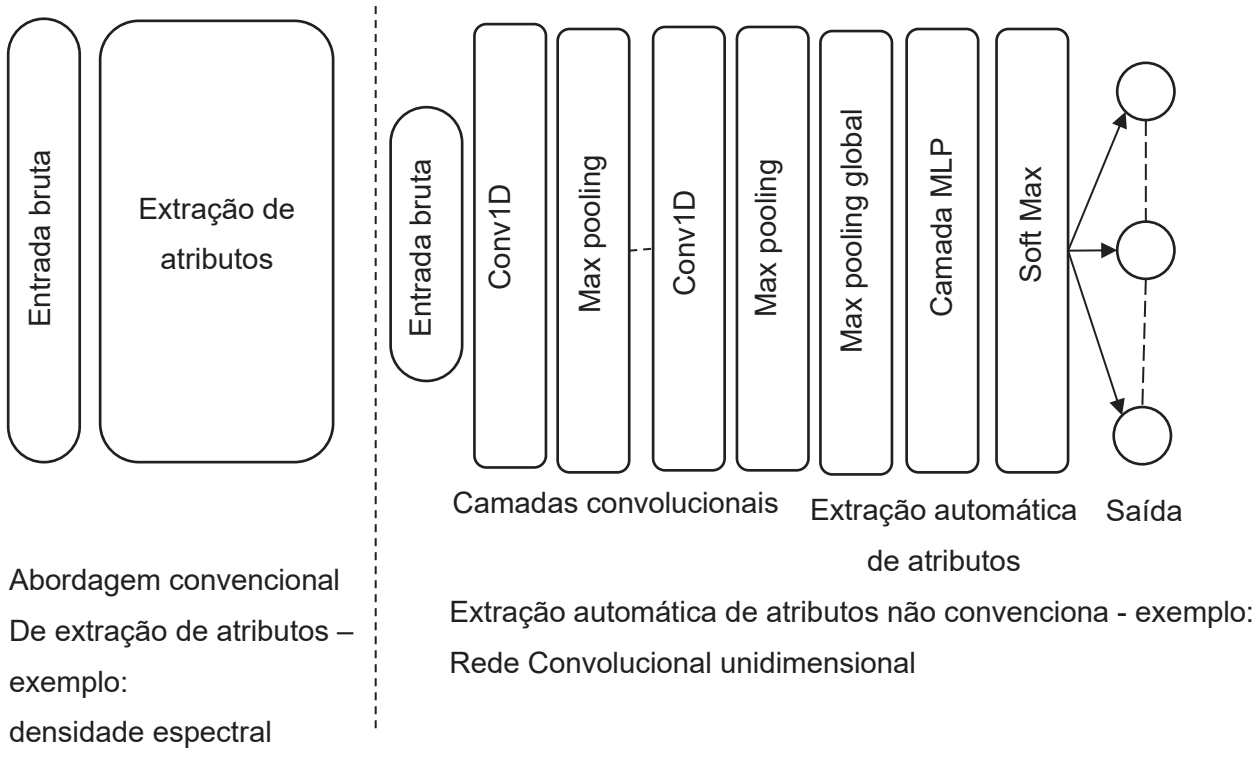
As Redes Neurais Profundas, ou em inglês, *Deep Neural Networks* (DNN), apresentam um número considerável de topologias e arquiteturas disponíveis em vários graus de complexidade (BOUWMANS et al., 2019). Dentre as arquiteturas comumente utilizadas destacam-se as aplicações em séries numéricas. As séries numéricas são genericamente um conjunto de dados estruturalmente organizados, em que a ordem dos elementos tem significado. Neste sentido, para Dhillon e Verma (2020), a arquitetura de redes neurais artificiais convolucionais foram responsáveis pelo rápido incremento no estado da arte do desempenho das redes neurais e pela consequente popularização desses métodos.

Por conseguinte, as redes neurais profundas em séries temporais usando a camada de Rede Neural de Convolução unidimensional (Conv1D) são empregadas como técnicas de extração de atributos dos sinais de entrada. Neste trabalho, adotou-se o viés da abordagem de engenharia de atributos como o equivalente das técnicas de extração de características. A engenharia de atributos é compreendida como o uso de técnicas de pré-processamento que visam aprimorar a visualização, filtrar redundâncias e determinar relações nos dados sob análise (KUHN; JOHNSON, 2019).

Qi et al. (2019) cunharam o termo, Extração Automática de Características, ou em inglês, *Automatic Feature Extraction* (AFE) usando a camada Conv1D. Eles mostraram uma grande melhora na precisão dos modelos de aprendizagem profunda avaliados. As camadas de Conv1D apresentam diversas vantagens, como realizar um aumento dimensional dos dados e o emprego da distorção temporal, ou seja, uma injeção de características de entrada de dimensão superior na camada totalmente conectada. A camada totalmente conectada é uma rede neural *Perceptron* Multicamadas, denominada em inglês como *Multilayer Perceptron* (MLP).

A FIGURA 7 mostra o processo de extração de características, que foi estratificado em duas etapas. A primeira, é a extração de características tradicionais conforme a seção 3.3.1. A segunda aplica os recursos das camadas Conv1D segundo o processo descrito na seção 3.3.2.

FIGURA 7 – MODELO DE ENGENHARIA DE ATRIBUTOS AUTOMATIZADO




FONTE: Adaptado de QI et al. (2019, p. 20).

Comparativamente, como indicado em Penge et al. (2018), a primeira etapa pode ser vista como uma representação dimensional dos dados de baixo nível e a

segunda como uma representação dimensional de alto nível. Especificamente, Kiranyaz et al. (2019) argumentaram que a camada Conv1D emula um analisador tempo-frequência, no sentido de capturar as distorções de frequência dentro do sinal sem pré-processamento. Consequentemente, a camada Conv1D aumenta a precisão em sistemas de reconhecimento de voz automático (PALAZ; MAGIMAI-DOSS; COLLOBERT, 2019).

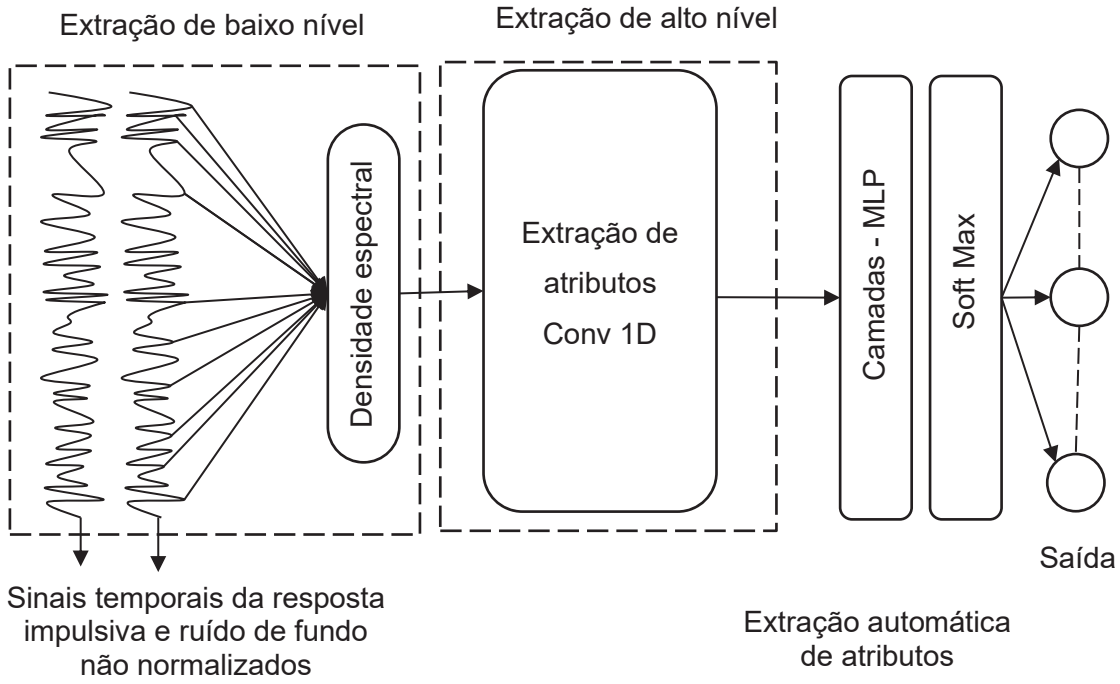
Além disso, tal abordagem, como afirmaram Mon, Pa-Pa e Thu (2018), possui alguns benefícios em relação à redução da complexidade do modelo. Logo, um modelo parcimonioso pode atingir uma precisão de última geração quando comparado com as abordagens tradicionais de extração de recursos, como a Transformada de Fourier de Curto Tempo, os coeficientes cepstrais de frequência de Mel e o emprego da transformada *wavelet* contínua.

Portanto, a camada Conv1D é particularmente útil para tarefas de classificação de dados sonoros brutos, sem a necessidade de avaliar os espectrogramas como largamente utilizados (LIU et al., 2019, PALAZ; MAGIMAI-DOSS; COLLOBERT, 2019, XIAO et al, 2020). Na próxima seção, definem-se as padronizações da nomenclatura e a formalização matemática para a criação do conjunto de entrada e alvo do treinamento

### 3.3.1 Padronização do conjunto de treinamento

Na seção 3.2.1, foi definido o pré-tratamento dos dados para a geração do banco de dados. Essa fase compreende a abordagem tradicional de engenharia de atributos. Estes atributos ainda se configuraram em baixo nível. Dessa forma, os dados de treinamento foram compostos por tuplas, que corresponderam a duas entradas e o alvo ordenados. As entradas foram a densidade espectral das respostas impulsivas e do ruído de fundo, conforme mostra a FIGURA 8.

FIGURA 8 – MODELO DE ENGENHARIA DE ATRIBUTOS COM A DISTINÇÃO DOS NÍVEIS




FONTE: Adaptado de Qi et al. (2019, p. 20).

Na FIGURA 8 evidencia-se o pré-processamento de uma única amostra de treinamento identificada por $s$. Verifica-se que a resposta impulsiva e o ruído de fundo no domínio do tempo são transformados para o domínio da frequência via densidade espectral. Em relação ao conjunto de treinamento, os dados de entrada foram normalizados aplicando-se a transformação via ***z-score,*** conforme a Eq. 26, qual seja,

$$x'_{m,i} = \frac{x_{m,i} - \mu(x_{m,i})}{\sigma(x_{m,i})} \tag{26}$$

em que $x'_{m,i} \in \mathbb{R}^{i \times s}$ é uma amostra de treinamento normalizada, $i$ , e $m \in \{1,2\}$ indica a entrada da amostra, no caso, para $m = 1$ corresponde à densidade espectral de potência da resposta impulsiva da sala e $m = 2$ identifica a densidade espectral de potência do ruído de fundo. Têm-se $s = 25000$ é a quantidade total de amostras de treinamento e $x_{m,i}$ são os dados não normalizados, os termos $\mu(x_{m,i})$ e $\sigma(x_{m,i})$ são a média e o desvio padrão das amostras de treinamento antes da normalização.

O conjunto de treinamento foi codificado como $\mathcal{X} = (X, Y)$ e expresso por meio da seguinte tupla $\mathcal{X}$, consolidada pela Eq. 27, a saber,

$$\mathcal{X} = (X, Y)_{s*b,j+k} = \left(\{x_{m,i,b}\}_{i=1}^{j}, \{y_{m,i,b}\}_{i=1}^{k}\right)_{m=1}^{s} \tag{27}$$

em que $s$ corresponde ao número de amostras, $b$ é o tamanho escalar do vetor de uma amostra, $m = \{1,2\}$ identificam as duas amostras de entrada $i$-ésimo par de entrada da $s$-ésimo . Por exemplo, para a amostra, $s = 1$, da tupla $\mathcal{X}$, com $m = 1$, o termo $\{x_{m,i}\}_{i=1}^{s}$ , expandido resulta na Eq. 28,

$$\{x_{m,i}\}_{i=1}^{s} = (x_{11}, x_{12}), \dots, (x_{12}, x_{m,i}), \tag{28}$$

em que é a $\{x_{m,i}\}_{i=1}^{s}$ entrada bidimensional para o número de amostra $s$.

Para cada amostra de treinamento $i$, a codificação "*one-hot*" foi usada para rotulagem das classes. A codificação "*one-hot*" infere que a rede neural profunda foi aplicada como um classificador. Portanto, $k$ de 1 até 10 representa a cardinalidade associada a classe da amostra $i$. Dessa forma, $k$ é a cardinalidade correspondente ao número de classes de saída dada por $\{y_i\}_{j=1}^{k} \in \{0,1\}$ que será usado na camada de saída SoftMax, a ser definida na seção 3.3.3.

O critério da adoção do número de classes, $k = 10,$ baseou-se no conceito da Diferença Minimamente Perceptível (DMP), que foi discutido na seção 2.5.1, como uma margem para o erro tolerável de medição. Com isto, infere-se uma extrapolação para definir a quantidade de classes, uma vez que, com o número de classes, $k = 10$, o ponto central é de 0,05, ou seja, maior que DMP e menor que 0,06, o que equivalente a duas vezes o DMP. Portanto, a rede será capaz de generalizar sobre a mínima diferença possível, sem gerar classes em excesso.

### 3.3.2 Extração de atributos via camada Conv1D

Após a extração das características de baixo nível realizada na seção 3.3.1, o sinal resultante foi direcionado para a primeira camada de entrada. Essa camada

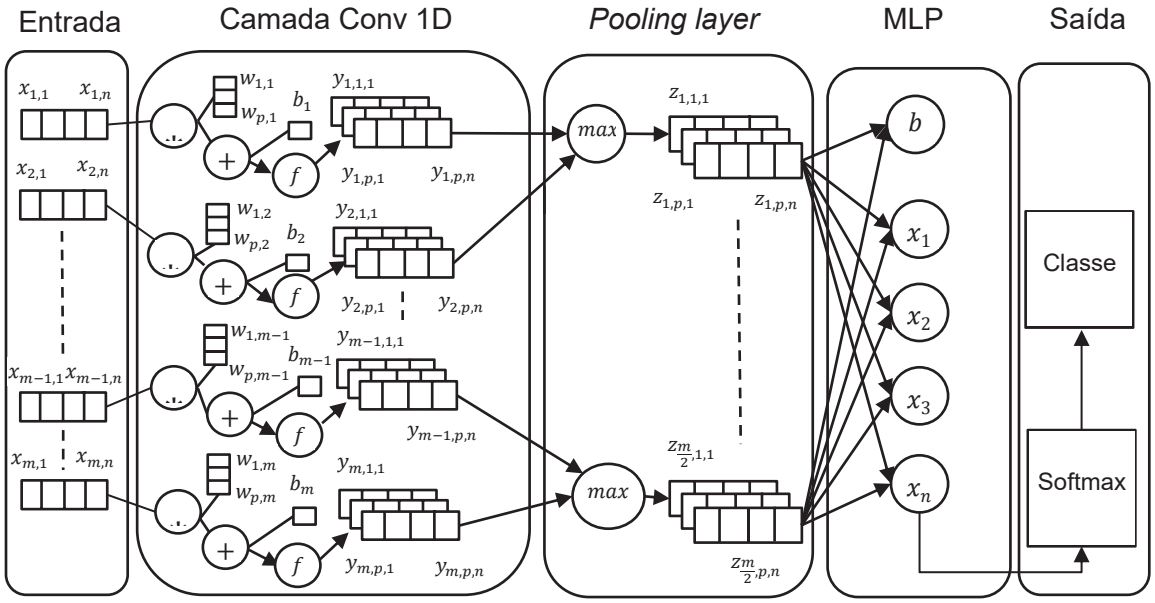
constitui o alto nível de extração de atributos de uma rede neural convolucional unidimensional. A convolução discreta aplica nessa camada é dada por

$$h_i(n) = (x * w)(n) = \sum_{j=1}^{m} w(j)x(i-k) \tag{29}$$

em que $*$ é a operação de convolução, $i$ é o número dos filtros de saída, onde $w$ é o kernel com comprimento $m$ é o número de filtros, $i-k$ é o comprimento da janela e $x$ é uma amostra de entrada (KHALIL et al, 2020; ZHAO et al., 2020).

A FIGURA 9 mostra a representação esquemática da camada de entrada e o contínuo processamento das amostras até a saída, a qual determina as classes.

FIGURA 9 – ESQUEMA DO SINAL DE ENTRADA DE UMA REDE NEURAL PROFUNDA CONVOLUCIONAL UNIDIMENSIONAL – 1D CONV NET




FONTE: Adaptado de Pan et. al (2018, p. 444).

A dimensão do tensor de saída da camada Conv1D é expressa em formato dimensional de um tensor 3-tupla, qual seja,

$$y_{i,j,k} = \sigma\left(\sum_{i=1}^{s} x_{i,k} * w_{j,i} + b_i\right) \quad (30)$$

em que $y_{i,j,k}$ é o mapeamento de saída da rede neural convolucional unidimensional, $\sigma$ é a função de ativação não linear que pondera a superposição de todas as unidades de kernel, $w_{j,i}$ é a matriz peso do kernel dada em relação a amostra de entrada $x_{i,k}$ com $1 \leq i \leq m$, e $b_i$ representa o termo de viés. As faixas de variação de $j$ e $k$ são, respectivamente $1 \leq j \leq m$ e $1 \leq k \leq m$ que são respectivamente o número de amostras de treinamento para um determinado lote e os números de kernel de convolução (PAN et al., 2018).

Isto posto, têm-se que a dimensão do comprimento da janela do kernel varia discretamente de $m = 1$ até $w$, dos números de filtros $n = 1$, até $i - k$, e do tamanho do lote de treinamento $u$, resultam num tensor de saída global, tal que $\{y_{i,j,k}\}_{i=1}^{s} = \left(\{u_i\}_{i=1}^{u}, \{w_i\}_{i=1}^{s}, \{n_i\}_{i=1}^{i-k}\right)_{i=1}^{s}$.

Com o objetivo de melhorar o desempenho na generalização, na saída da camada da Conv1D realizou-se uma filtragem para diminuir a quantidade de amostras, ou seja, fez-se um processo de *downsampling* amostral via *max-polling*. O filtro denominado de *max-polling* considera apenas os elementos com a maior magnitude no tensor, $y_{i,j,k}$, de cada filtro $k$, para valores associados na saída da camada anterior. Assim,

$$z_{i,j,k} = max_{k=1}^{r}(y_{i-1,j,k}) \quad (31)$$

em que $z_{i,j,k}$ é a saída máxima da camada *max-polling*, $y_{i-1,j,k}$ é o peso da camada anterior, e $max$ é o operador matemático máximo. Esse tipo de camada aumenta o desempenho do treinamento, pois os efeitos de pequenos pesos são minorados pela otimização do gradiente (PAN et al., 2018).

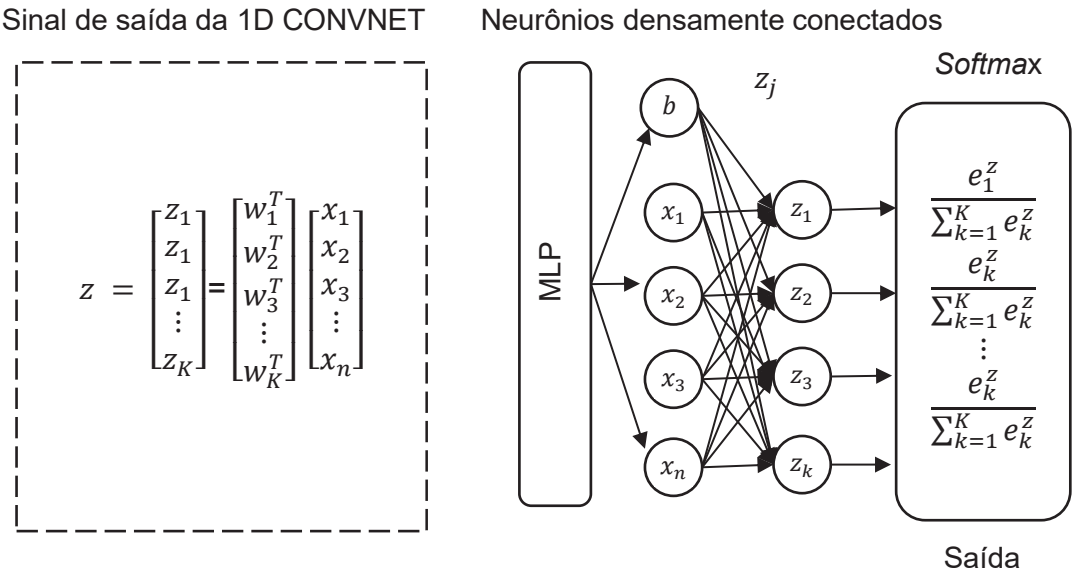
De forma análoga, a camada de *max pooling global* avalia o tensor $z_{i,j,k}$ e calcula a média nas direções das dimensões de $i, j$ e $k$, resultando em um vetor unidimensional. O vetor é dado por $\{\tilde{x}_i\}_{i=1}^{n} = \max \text{pooling global}\,(z_{i,j,k})$. O sinal, $\{\tilde{x}_i\}_{i=1}^{n}$

é alimentado a posteriori para as camadas densamente conectadas (MLP) e estas para a função Softmax, esse procedimento será tratado na seção 3.3.3.

### 3.3.3 Ajustes dos pesos via algoritmo de retropropagação de erros

As atualizações iterativas dos pesos na camada Conv1D são realizadas via algoritmo de retropropagação de erros modificado (RUMELHART, HINTON, WILLIAMS, 1986). Para tanto, deve-se ponderar a interação com a camada densamente conectada. Ao final, os pesos são comutados na camada densamente conectada. Essa camada possui a unidade básica de processamento, ou seja, os neurônios artificiais do tipo *perceptron*. Com isso, a FIGURA 10 mostra a representação expandida da camada densamente conectada com suas respectivas unidades de processamento.

FIGURA 10 – MODELO DA CAMADA DENSAMENTE CONECTADA E SAÍDA SOFTMAX




FONTE: Adaptado de Zhang et al. (2022, p. 3610).

Sem perda de generalidade, na função de saída *softmax* da FIGURA 10, adotou-se o parâmetro $\theta$ como o parâmetro de otimização global para a função $\mathcal{L}$. A camada de saída SoftMax realiza uma comparação entre as classes alvo e as classes que foram preditas. Retorna-se, assim, a probabilidade condicional, dado um conjunto

de pesos, na camada densamente conectada, e estima-se a pertinência, ou seja, a probabilidade de a amostra de entrada pertencer a classe de referência, conforme mostrado na Eq. 32, dada por,

$$h_\theta\left(x^{(i)}\right) = \begin{bmatrix} p\left(y^{(i)} = 1|x^{(i)};\theta\right) \\ p\left(y^{(i)} = 2|x^{(i)};\theta\right) \\ \vdots \\ p\left(y^{(i)} = 1|x^{(i)};\theta\right) \end{bmatrix} = \frac{1}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_l^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix} \tag{32}$$

em que $h_\theta\left(x^{(i)}\right)$ é a distribuição de probabilidade para a obtenção da classe $y^{(i)}$ dada a amostra de entrada $x^{(i)}$. A classe prevista é calculada como o argumento máximo das probabilidades condicionais, dadas por $\arg\max_{0 \le x \le k} h_\theta\left(x^{(i)}\right)$.

Na FIGURA 10, assume-se que os valores de vetor $x$ foram obtidos de um *max pooling global*, conforme evidenciado na seção 3.3.2. Ao avaliar o ajuste dos pesos da camada Conv1D, uma alusão é feita ao modelo tradicional da saída de um neurônio do tipo perceptron multicamadas propostas originalmente por Rumelhart, Hinton e Williams (1986).

Nesse quesito, Abdeljaber et al. (2017) apresentaram uma variante do modelo de retropropagação dos erros, aplicando a abordagem dos filtros convolucionais. Concomitantemente, Kiranyaz, Ince e Gabbouj (2015), Liu et al. (2017) Wang et al. (2019) consolidaram um modelo de otimização via gradiente descendente estocástico para o ajuste dos pesos, sendo que a função objetivo utilizada foi o erro médio quadrático.

Este trabalho adotou a função objetivo atribuída para a atualização dos pesos na camada Conv1D conforme Zeng et al. (2014). Já a atualização dos pesos na camada densamente conectada seguiu a abordagem tradicional de retropropagação dos erros conforme Rumelhart, Hinton e Williams (1986). Assim sendo, a interação do passo de alimentação a frente na camada Conv1D é dada pela Eq. 33 (ZENG et al., 2014), com

$$w_{i,j} = \sum_{j}^{s} w_{j,i}^{l-1} \sigma\left(z_{i,j,K}\right) + w_i^{l-1} \tag{33}$$

em que a interação dos pesos é

$$w_{i,j} = w_{j,i}^{l-1} - \alpha \frac{\partial \mathcal{L}_i}{\partial w_{i,j}} \tag{34}$$

sendo o gradiente dado como

$$\frac{\partial \mathcal{L}_i}{\partial w_{i,j}} = \sum_{i=1}^{C} t_i \frac{\partial \mathcal{L}_i}{\partial w_{i,j}} \tag{35}$$

em que $w_{i,j}$ são os pesos da saída do filtro convolucional unidimensional ponderando-se as funções de *max-polling* e *max pooling global*, e $\mathcal{L}$ é a função objetivo a ser otimizada.

A camada densamente conectada (MLP) foi configurada com o número de unidades iguais a $k$ classes. Adotou-se a função objetivo, $\mathcal{L}$, como a entropia cruzada categórica, conforme a Eq. 36, (KINGMA; BA, 2014, CHOLLET et al., 2018; CHOLLET, 2018), a saber,

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{C} \mathbf{1}(y_i) \frac{e^{w_k^T b_i}}{\sum_{j=1}^{C} e^{w_k^T b_i}} \tag{36}$$

em que $\mathbf{1}$ representa a matriz com todas as entradas iguais a unidade, $y_i$ é a classe real de uma amostra, $C$ é o número de classes, e $N$ é quantidade de neurônios do tipo perceptron na saída da camada MLP que precede a camada de saída SoftMax. A otimização foi realizada de acordo com o algoritmo Adam, para a atualização dos pesos (CHOLLET et al., 2018). O termo Adam não é um acrônimo, ele representa em inglês como, *adaptive moment estimation*, e foi o nome dado ao algoritmo que realiza otimização estocástica ao empregar apenas gradientes de primeira ordem (KINGMA; BA, 2014).

### 3.3.4 Critério de validação dos modelos propostos

A validação dos modelos em aprendizagem profunda requer um balanço simbiótico entre a qualidade do ajuste e a generalização. A qualidade do ajuste refere-se ao uso de métricas que correlacionam a quantidade de acertos e a quantidade de falsos positivos no modelo. A generalização refere-se à qualidade das estimativas do modelo de aprendizagem, quando este é apresentado a dados fora do espaço amostral de treinamento.

Para Schelter, Rukat, Biessmann (2020), deve-se observar o viés do erro, se é do tipo I ou tipo II, para realizar o teste de significância estatística. Para tanto, uma tabela da verdade deve ser confeccionada. Usualmente em redes neurais artificiais profundas, aplica-se o método da validação cruzada, *k-cross validation*, para avaliar a qualidade do estimador. Para tanto, neste trabalho, adotaram-se dois conjuntos, treinamento e teste. A proporção da alocação da quantidade de amostras nestes conjuntos foi de 70% e 30%, respectivamente. Ainda deve-se atentar para o desempenho do modelo para dados não apresentados durante o treinamento.

No contexto deste trabalho, empregaram-se as redes neurais artificiais profundas com um objetivo de realizar uma classificação sobre os dados da densidade espectral da resposta impulsiva e do ruído de fundo. Para tanto, adotou-se a abordagem da criação da matriz de confusão e desta avaliou-se o valor da acurácia, $ACC$, e o $F1-score$. A acurácia é dada pela Eq. 37, com

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{37}$$

em que $ACC$ é a acurácia do modelo (parâmetro representa o desempenho global do estimador), a $ACC$ é uma métrica que representa o desempenho global do estimador e é calculada como a proporção de previsões corretas ($TP + TN$) em relação ao total de previsões ($TP + TN + FP + FN$). Portanto, a acurácia é uma medida geral de quão bem o modelo está fazendo previsões corretas em relação a todas as previsões feitas. $TN$ (*True Negatives* - Verdadeiros Negativos) representa a quantidade de casos em que o modelo previu corretamente uma classe negativa (por exemplo, a ausência de uma condição) e estava correto em sua previsão, $FP$ (*False Positive*s - Falsos

Positivos), representa a quantidade de casos em que o modelo previu erroneamente uma classe positiva (por exemplo, a presença de uma condição) quando, na verdade, a classe real era negativa; $FN$ (*False Negatives* - Falsos Negativos) representa a quantidade de casos em que o modelo previu erroneamente uma classe negativa quando, na verdade, a classe real era positiva. Ou seja, o modelo não conseguiu identificar corretamente casos positivos e $TP$ (*True Positives* - Verdadeiros Positivos): Isso representa a quantidade de casos em que o modelo previu corretamente uma classe positiva e estava correto em sua previsão. São os casos em que o modelo identifica corretamente a presença da condição ou evento de interesse. O $F1 - score$ é particularmente útil pois informa o balanço entre a acurácia e a frequência de acerto em uma determinada classe. O $F1 - score$ é dado por

$$F1 - score = 2 \times \frac{\frac{TP}{TP + FP}\frac{TP}{TP + FN}}{\frac{TP}{TP + FP} + \frac{TP}{TP + FN}} \tag{38}$$

ainda no contexto da validação cruzada, apenas uma topologia pode ser insuficiente para determinar um modelo (estimador) que melhor se ajusta aos dados. Nesse caso, empregaram-se técnicas de otimização dos hiperparâmetros em modelos de redes neurais profundas.

Wu, Chen e Liu (2020) fornecem uma formulação formal do problema de otimização dos hiperparâmetros. Para tanto, explicita-se um modelo de aprendizagem profunda, representado por $\mathcal{A}$, modelo este que possui os hiperparâmetros, $\lambda$. Logo, identifica-se o modelo hiperparametrizado como $\mathcal{A}_\lambda$. Ao realizar uma correspondência univariada de um espaço vetorial discreto para os hiperparâmetros, este fica subentendido como $\lambda \in \Lambda$. Por consequência, objetiva-se estipular uma configuração teórica para o espaço vetorial da combinação ótima dos hiperparâmetros $\lambda^*$ que resultam no modelo de referência, denotado por $\mathcal{D}$, em que se minimiza a diferença entre o melhor modelo possível e o modelo obtido. A estimativa do modelo ótimo é

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \mathbb{E}_{(D_{treino}, D_{validação}) \sim D} \mathcal{L}(\mathcal{A}_\lambda, D_{treino},) D_{validação} \tag{39}$$

em que $\lambda^*$ representa o conjunto ótimo de hiperparâmetros, e $\mathcal{L}(\mathcal{A}_\lambda, D_{treino},)D_{validação}$ é a função objetivo do modelo de aprendizagem profunda do modelo $\mathcal{A}$, (ver Eq. 35). Os termos $D_{treino}$ e $D_{validação}$ são os conjuntos de treino e validação, respectivamente, os quais seguem a divisão proporcional do método da validação cruzada.

Nota-se que, ao adotar a abordagem da Eq. 39, o problema corresponde à otimização da acurácia dos modelos. Como corolário, realiza-se a maximização das métricas de ajuste da qualidade do modelo, conforme a Eq. 37 e a Eq. 38. A TABELA 3 mostra as topologias avaliadas para o emprego da otimização de hiperparâmetros. Salienta-se que foram utilizadas validações cruzadas (CV = 10).

TABELA 3 – TOPOLOGIA DA REDE CONVOLUCIONAL APLICADA NA VALIDAÇÃO DO MVPAnP

| Camada | Nome | Função de Ativação | Hiperparâmetro |
|---|---|---|---|
| 1 | Camada de entrada | # | Tamanho do lote: 32, 64 e 128 |
| 2 | Conv1D | ReLu | Tamanho do kernel: 2, 8 e 16<br>Número de filtros: 2, 8, 16 |
| 3 | Conv1D | ReLu | Tamanho do kernel: 2, 8 e 16<br>Número de filtros: 2, 8, 16 |
| 4 | *MaxPooling1D* | # | # |
| 5 | Conv1D | ReLu | Tamanho do kernel: 2, 8 e 16<br>Número de filtros: 2, 8, 16 |
| 6 | Conv1D | ReLu | Tamanho do kernel: 2, 8 e 16<br>Número de filtros: 2, 8, 16 |
| 7 | *Global average pooling* | # | # |
| 8 | *Dropout* | # | Razão de corte: 0,1; 0,25 e 0,50 |
| 9 | Camada MLP(*1) | ReLu | Número de neurônios: 10; 100 e 200 |
| 10 | Camada MLP | ReLu | Número de neurônios: 10; 100 e 200 |
| 11(a) | Saída do MLP – Sem TF | Softmax | 10 |
| 11(b) | Saída do MLP – Com TF | Softmax + MSE/MVPAnP (*2) | 10 |

FONTE: O autor (2023).

NOTA: (*1) Nota para desambiguação de termos: MLP é a camada *multilayer perceptron*, no módulo Keras, essa camada recebe a nomenclatura de FC, ou seja, do inglês *Fully Connected Layer*, ou ainda *dense layer.*

NOTA: (*2) O algoritmo MVPAnP é utilizado como função de perda customizada e representa a injeção da transferência de aprendizagem profunda a um modelo padrão, nesse caso foi a rede 1D CONV NET. A apresentação da implementação do algoritmo MVPAnP é o objetivo desta tese, portanto consultar a seção 3.6 para mais detalhes.

Na TABELA 3, observa-se a alocação de camadas duplas, para as camadas de Conv1D e Camada MLP. Isso se deve, ao aumento intrínseco da capacidade de generalização (ZHAO et al., 2020). Postula-se que tal efeito advém do princípio da superposição e do aumento da complexidade da representação de uma maior

quantidade de hiperplanos no espaço de aprendizado, segundo o teorema da aproximação universal (CYBENKO, 1989). Com isso, a rede neural tende a especificar e especializar o espaço latente para atributos dos sinais de entrada. Espaços latentes são representações abstratas de dados usados em aprendizado de máquina, permitindo a extração de características relevantes e a redução da complexidade. Eles são amplamente aplicados em áreas como processamento de linguagem natural e visão computacional.

Com base na otimização dos hiperparâmetros, realizou-se uma busca de malha uniforme para identificar a melhor combinação destes que minimiza a Eq. 39. Adicionalmente, usou-se a média de 10 treinamentos de cada modelo de forma independente e avaliaram-se os resultados das métricas de qualidade, no caso, a acurácia $ACC$ e o fator $F1 - score$.

Abordagens similares à adotada neste presente trabalho foram empregadas recorrentemente na literatura (HUTTER; KOTTHOFF; VANSCHOREN, 2019, PAN et al. 2018, LI et al., 2019, CRUCIANI et al., 2020, LU et al., 2020, ZHAO et al. 2020).

## 3.4 TRANSFERÊNCIA DE APRENDIZAGEM PROFUNDA

Ao analisar as heurísticas de aprendizado profundo, pode-se almejar reimplementar um modelo de redes neurais artificiais que foi previamente ajustado para um conjunto de dados específico, para ser aplicado em outro. Para tanto, esses novos conjuntos de dados devem ter uma estrutura topológica semelhante, ou devem manifestar, no espaço das probabilidades, uma razão de verossimilhança relativamente alta. Estatisticamente, tal problema dá-se nas relações entre os ditos domínios de origem e de destino (alvo). A afinidade entre eles pode ser compreendida por regras de aprendizagem baseadas em projeções (ZHUANG et al., 2019).

Nesse sentido, o domínio fonte é mapeado para o domínio alvo por meio de um operador de dimensão superior que permite o compartilhamento do conhecimento entre os modelos devidamente treinados. Através de tal *proxy*, uma descrição estatística aprimorada para o domínio alvo é obtida.

Comumente há estudos de caso de engenharia em que é impraticável desenvolver um modelo de aprendizagem profunda diretamente. Isso pode ocorrer por motivos diversos, no entanto, um dos mais importantes é a impossibilidade de

gerar amostras de treinamento suficientes capazes de produzir uma generalização satisfatória para a emprego adequado desses modelos (HARRIS, 1991, LI; GRANDVALE; DAVOINE, 2020). Quando tais casos emergem é mais adequado desenvolver outra heurística de treinamento/inferência, como por exemplo, a transferência de aprendizagem, geração de dados sintéticos dentre outras heurísticas, que consigam cobrir essa lacuna na aquisição de amostras de treino.

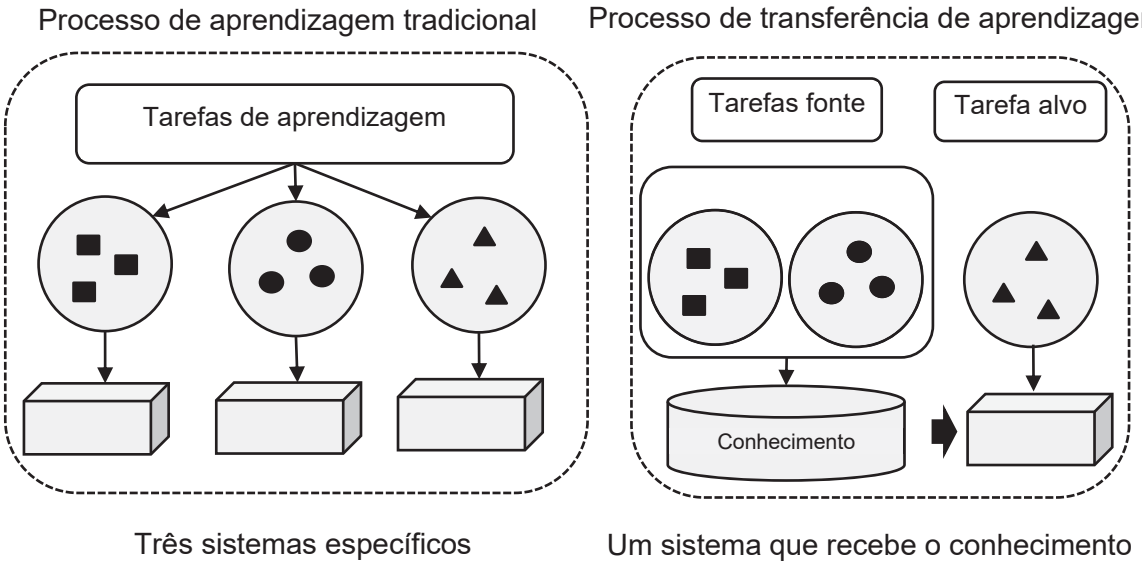
O fenômeno previamente mencionado é especialmente verdadeiro no campo da acústica, especialmente quando há fortes não linearidades. Isso ocorre quando a saída de um sistema pode se comportar de maneira completamente diferente, dadas pequenas variações em suas condições iniciais e de contorno (SAARELMA et al., 2016).

### 3.4.1 Definição de transferência de aprendizagem

Em termos gerais, a transferência de aprendizagem profunda refere-se a um processo que reimplementa uma rede neural profunda para realizar uma nova tarefa de aprendizado estatisticamente similar. Da mesma maneira, essa reimplementação objetiva melhorar uma certa métrica de desempenho anterior em relação às amostras de treinamento invisíveis, ou seja, amostras que não foram apresentadas no treinamento do primeiro modelo. A melhora do desempenho é alcançada devido ao “conhecimento” adquirido de outro modelo, previamente treinado dentro de uma tarefa estatisticamente semelhante.

Para contextualizar a aprendizagem por transferência, é necessário estabelecer uma notação em relação para os domínios de origem e o de alvo. O domínio de origem armazena o estado do conjunto de dados de maneira estruturalmente organizada. Em geral, a estrutura topológica é caracterizada por um espaço de distribuição probabilística bem definida. Nesse sentido, para estabelecer a transferência de aprendizagem, definem-se dois domínios representados por espaços vetoriais. A FIGURA 11 mostra a comparação entre as abordagens tradicionais e a transferência de aprendizagem.

FIGURA 11 – DIFERENCIAÇÃO DA APRENDIZAGEM DE MÁQUINA TRADICIONAL E DA TRANSFERÊNCIA DE APRENDIZAGEM




FONTE: Adaptado de Pan e Yang (2009, p. 1346).

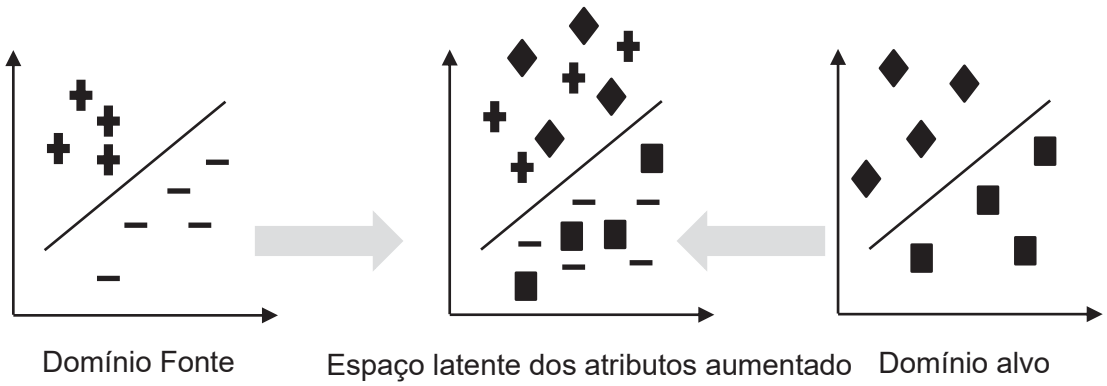
Observa-se na FIGURA 11(a), que um modelo tradicional é especificado para um objetivo de aprendizagem, referenciado como atividade para um banco de dados específico. Ao contrário, na FIGURA 11(b), observa-se que os dados são agrupados e um objetivo de aprendizagem é especificado para o modelo de aprendizagem. Em seguida, aplicando as regras de aprendizagem, transfere-se parte do conhecimento para outro modelo de aprendizagem.

Nota-se que transferência pode ser dada pelo uso dos pesos sinápticos previamente ajustados. Normalmente, em aplicações práticas, a escassez de dados pode sugir, limitando o uso dos modelos de aprendizagem profunda. Nesses casos, a transferência de aprendizagem é particularmente eficiente.

O domínio de origem, ou domínio fonte, é dado por $\mathcal{D} = \{\mathcal{X}, \mathcal{P}(\mathcal{X})\}$, em que $\mathcal{X}$ contém o espaço dos atributos dados por $\{x_{m,i}\}_{i=1}^{j=2}$. No caso, em $\mathcal{X}$, realiza-se algum procedimento de engenharia de atributos(ver seção 3.3). O termo $\mathcal{P}(\mathcal{X})$ indica uma distribuição de probabilidade marginal de $\mathcal{X}$. Já, o domínio alvo é dado por $\mathcal{T} = \{\mathcal{Y}, \mathcal{f}\}$, em que $\mathcal{Y}$ é o espaço vetorial correspondente ao classificador, e $\mathcal{f}$ representa sua respectiva função de distribuição condicional probabilística, que foi aprendida pelo classificador. No contexto deste trabalho, $\mathcal{f}$ torna-se a função Softmax (ver seção

3.3.3). Graficamente a FIGURA 12 demonstra o domínio alvo, $\mathcal{T}$, que codifica um rótulo de uma classe, $\mathcal{K}$, desconhecida, correspondendo a $\{y_i\}_{i=1}^{k} \in \{0, +1\}$ de uma amostra de domínio de origem.

FIGURA 12 – HIPERPLANO DA FRONTEIRA DE DECISÃO NO ESPAÇO LATENTE




FONTE: Adaptado de Weiss, Khoshgoftaar e Wang (2016, p. 24).

Especificamente, a função $\not{f}$ atribui explicitamente uma probabilidade condicional de uma amostra de $\mathcal{Y}$ extraída de $\mathcal{T}$, pertencer a uma certa classe $\mathcal{y}_{\not{k}}$ , com base no conhecimento extraído do domínio fonte, tal que,

$$\not{f}(x_j) = \{\mathcal{P}(\mathcal{y}_{\not{k}}|x_j)|\mathcal{y}_{\not{k}} \in |\mathcal{Y}, \mathcal{K} = 1, \cdots, |\mathcal{Y}|\} \tag{40}$$

Desse modo, a transferência de aprendizagem melhora a estimativa da função de decisão, $\not{f}$, usando uma porção da informação pela observação sobre uma amostra de treinamento dos pares $\{(\mathcal{D}_{s_i}, \mathcal{T}_{s_i})|i = 1, \cdots, m^s\}$, onde $m^s$ é uma amostra do domínio fonte. Consequentemente, com os recursos do domínio fonte, estima-se o rótulo da classe $\mathcal{K}$ no domínio alvo, $\mathcal{T}$, exclusivamente com base no conhecimento estatístico contido em $\mathcal{D}$.

Alusivamente, Eq. 41 mostra a função objetivo genérica aplicada na transferência de aprendizagem (POELITZ, 2014), com

$$\mathcal{L}(p_s(x), p_t(x)) = D(p_s, p_t) + \lambda d(h_s, h_t) \tag{41}$$

em que $\mathcal{L}$ é a função objetivo, $D$ é uma função de discrepância estatística entre o domínio fonte $p_s$ e o domínio alvo $p_t$, $\lambda$ é um parâmetro regularizador e $d$ estima a distância entre as duas distribuições, realizando um teste de hipótese de pertencimento de uma amostra pertencer entre as classes $h_s$ e $h_t$.

Segundo Zheng et al. (2019), as funções objetivo de classe bayesianas para $\mathcal{L}$ são boas candidatas, pois essas podem ser usados para condições de varredura no espaço de soluções viável convexo, considerando o efeito a priori da estimativa e o comuta com a estima a posteriori. Contudo, essas apresentam limitações ao ajuste, quanto aos tipos de distribuições probabilísticas dos dados de treinamento. Por exemplo, se as distribuições do domínio de origem em relação ao domínio de destino (alvo) diferem significativamente, elas provavelmente produzirão resultados espúrios, que podem levar a uma classificação incorreta do domínio alvo (RABIN et al., 2019).

## 3.5 PARADIGMA DA TRANSFERÊNCIA DE APRENDIZAGEM PROFUNDA

Essa seção objetiva definir a formulação para os métodos projetivos de aprendizagem por transferência e estabelecer as bases para o desenvolvimento de um novo tipo de modelo de transferência de aprendizagem.

### 3.5.1 Transferência de aprendizagem baseada em projeção em espaços latentes

Após findar a seção 3.4, verificou-se que os modelos de transferência baseados em projeções e medição do distanciamento intraespecífico das projeções são bons candidatos para a função custo da Eq. 41. Conforme definido previamente, a transferência de aprendizagem depende de uma hipótese da similaridade estatística, para gerar previsões afins. A inferência estatística é gerada sobre um conjunto de dados invisíveis durante o treinamento. Por extensão, as classes não rotuladas são estimadas para o domínio alvo, $\mathcal{T}$, com base no conhecimento a priori do domínio fonte, $\mathcal{D}$.

Como resultado, a suposição de similaridade estatisticamente significativa entre o domínio de origem e de alvo deve coexistir sob alguma extensão mensurável. Para tanto, uma variante da Eq. 41 surge, ponderando o efeito do distanciamento no

espaço das distribuições. Do mesmo modo, o efeito da variância das classes é considerado simultaneamente, conforme

$$\min_{\Phi} \frac{\big(DIST(\mathcal{D},\mathcal{T};\Phi) + \lambda\Omega(\Phi)\big)}{VAR(\mathcal{D}\cup\mathcal{T};\Phi)} \tag{42}$$

em que $\Phi$ é um operador que realiza um mapeamento de redução dimensional, $DIST(\cdot)$ representa uma métrica para o distanciamento de distribuições, $\Omega(\Phi)$ é uma função regularizadora e $VAR(\cdot)$ é a variância das amostras dos domínios $\mathcal{D}$ e $\mathcal{T}$. Nestes domínios, subentende-se a presença de dados categóricos (XU et al. 2019).

Uma comparação entre a função objetivo generalizada para a transferência de aprendizagem, Eq. 41, e a Eq. 42 revela que, ao adaptar uma abordagem de projeção em baixas dimensões, ou ao utilizar espaços latentes, pode-se obter uma descrição estatística robusta dos dados. No entanto, deve-se atentar para quando a não afinidade surge entre as distribuições marginais dos domínios de origem e de alvo.

Para lidar com essa questão, introduziu-se a métrica denominada de Discrepância Média Máxima, em inglês, *Maximum Mean Discrepancy* (MMD). Nesse sentido, a Eq. 43 foi uma das primeiras tentativas de formular, de uma maneira semiempírica, um método de distribuição-distância para expressar a similaridade estatística das amostras provenientes do subespaço de treinamento. Tem-se,

$$MMD(X^S, X^T) = \left\| \frac{1}{n^S}\sum_{i=1}^{n^S} \Phi\big(x_i^S\big) - \frac{1}{n^T}\sum_{j=1}^{n^T} \Phi\big(x_i^T\big) \right\|_{\mathcal{H}}^2 \tag{43}$$

em que $n^S$ e $n^T$ são as quantidades de amostras nos domínios da fonte e alvo respectivamente e $\Phi(\cdot)$ é uma função que transforma o espaço vetorial das amostras para uma dimensão no espaço latente. Essa função pode ser interpretada como uma classe de funções de engenharia de atributos. Já $\|\cdot\|$ indica a norma métrica no espaço $\mathcal{H}$. O espaço $\mathcal{H}$ corresponde ao espaço de Hilbert reproduzido por Kernel.

Weiss e Khoshgoftaar (2016) forneceram métodos alternativos para a transferência de aprendizagem. Esses métodos baseiam-se em projeções, usando um mapeamento não linear genérico. Em relação aos métodos tradicionais, Zheng et

al. (2019) e Zhuang et al. (2019) estratificaram uma divisão do sentido de aprendizagem baseada em projeção. Assim, sendo Zhuang et al. (2019) consolidaram os métodos de Análise de Componentes Principais do Kernel (KPCA), da rede autocodificadora variacional (VAE), além da Discrepância Média Máxima (MMD).

### 3.5.2 Análise de componentes principais por núcleo

A Análise de Componentes Principais por Núcleo, ou, em inglês, *Kernel Principal Component Analysis* (KPCA) é uma variante transcendental do método da Análise de Componentes Principais (ACP) (JOHN; NELLO, 2004). Wibowo (2018) propõe a representação dos dados de treinamentos de forma compacta como $\left(y_i, x_{i1}, x_{i2}, \cdots, x_{ip}\right) \in \mathbb{R}^{p+1}$, para $i = 1, 2, \cdots, N$. Para simplificar a notação, define-se o vetor $\{x_i\}_{i=1}^{N} = \left(x_{i1}, x_{i2}, \cdots, x_{ip}\right)^T$, onde $(\cdot)^T$ é o operador de transposição matricial. O KPCA realiza o mapeamento via uma função genérica $\Psi$ aplicada em $\{x_i\}_{i=1}^{N}$, que realiza uma transformação para uma dimensão superior no espaço dos atributos. O espaço dos atributos é representado por $\mathcal{F}$, tal que $\Psi: \mathbb{R}^p \rightarrow \mathcal{F}$. A transformação equivalente é dada por

$$\Psi = (\Psi(x_1)\ \Psi(x_2)\ \cdots\ \Psi(x_N)\ )^T \tag{44}$$

em que $\Psi$ possui dimensão $N$ x dim($\mathcal{F}$), onde dim($\mathcal{F}$) é a dimensão do espaço dos atributos, dado por $p_{\mathcal{F}}$, sendo que, $p_{\mathcal{F}} > p$ . Por consequência, estima-se a matriz de variância semiempírica, $\hat{\mathbb{C}}$, por

$$\hat{C} = \frac{1}{N}\sum_{i=1}^{N} \Psi\,(x_i)\Psi(x_i)^T = \frac{1}{N}\Psi^T\Psi \tag{45}$$

em que $\hat{\mathbb{C}}$ possui dimensão $p_{\mathcal{F}} \times p_{\mathcal{F}}$.

Originalmente, Rosipal e Trejo (2001) e Rosipal et al. (2001) mostraram que a forma de $\Psi$ não precisa ser necessariamente conhecida. Para tanto, emprega-se um artifício matemático denominado de truque do Kernel. Este truque permite a diagonalização de $\mathrm{k}$, por meio dos autovalores do problema equivalente de $\lambda u = \hat{\mathbb{C}}u$,

onde $u$ representam os autovetores e $\lambda$ autovalores de $\hat{\mathbb{C}}$. A diagonalização via kernel fica definida como,

$$k = \Psi^T \Psi \tag{46}$$

O kernel, k, possui dimensão de $N \times N$, sendo o problema, então, transformado para,

$$k\tilde{u} = n\lambda\tilde{u} \tag{47}$$

onde os autovetores, $\{u^k\}_{k=1}^N$, são dados por

$$u^k = (n\lambda_k)^{-1/2} \Psi^T \tilde{u}^k = \tilde{\lambda}_k \Psi^T \tilde{u}^k \tag{48}$$

em que $\tilde{u}^k$ é a estimativa do autovalor obtido pela solução da Eq. 47. A projeção do autovetor ortogonal sobre os dados originais é dada por,

$$\beta_k(x) = \Psi(x_i)^T u^k = \tilde{\lambda}_k^{\ -1/2} \sum_{i=1}^{n} u_i^k K(x_i, x) \tag{49}$$

Para Shawe-Taylor e Cristianini (2004), em implementações computacionais, a solução padrão do problema de autovetores numérica pode ser aplicada diretamente na Eq. 47. Por meio da projeção $\beta_k(x)$ sobre os autovetores ortogonais, obtém-se uma descrição espacial das classes para um problema de transferência de aprendizagem. As funções de transformação kernel serão apresentadas na seção 3.6.

### 3.5.3 Redução dimensional via autocodificadores variacionais

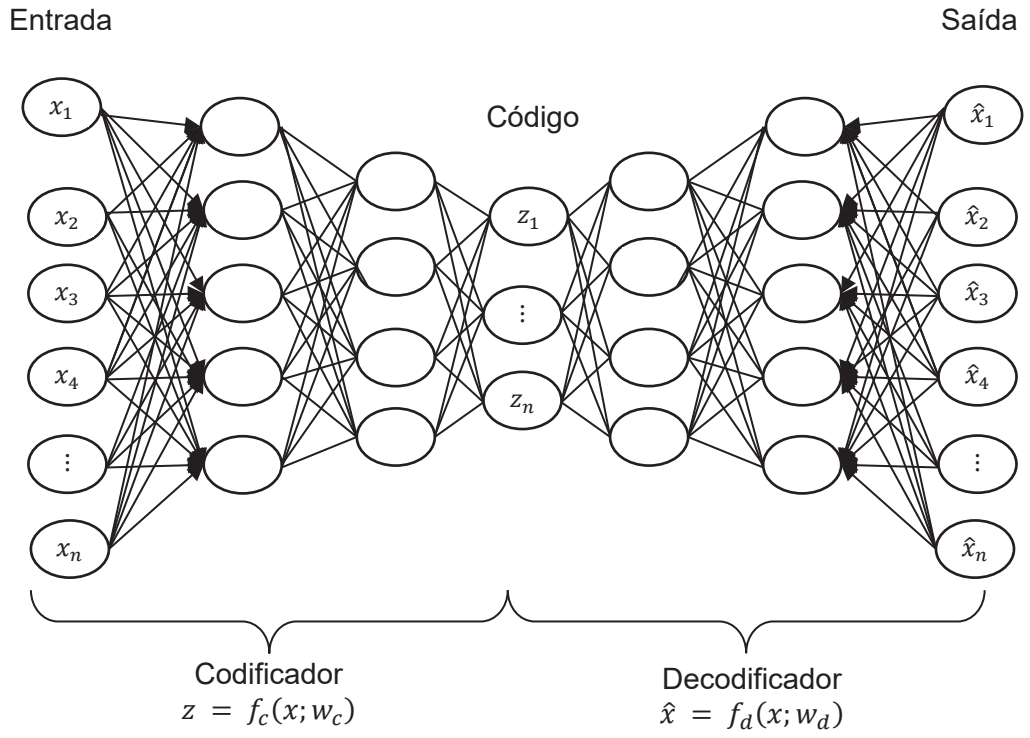
Na seção 3.4.1, mencionou-se que, segundo Zheng et al. (2019), as funções bayesianas não necessariamente são ótimas candidatas para os problemas de transferência de aprendizagem. Por outro lado, Rezende, Mohamed e Wierstra (2014) aludiram que a formulação híbrida entre o método variacional bayesiano e as redes neurais podem ser particularmente úteis. Essa utilidade é obtida pela transformação

de um problema inferencial estatístico multidimensional em um problema de otimização numérica convexa.

Nessa transmutação de domínios, na otimização numérica, as informações a priori das amostras são ponderadas. Com isso, utiliza-se a descrição estatística de uma amostra como um rastreador invariante no espaço amostral das classes. Nessa conjuntura, as redes neurais autocodificadoras, conhecidas como *autoencoders*, são responsáveis por um tipo de mapeamento funcional identificado pela autorrepresentação dos dados via uma função identidade.

Como isso, a meta-representação dos dados produz uma generalização, na medida que, ao aplicar um artifício funcional, dado por: $f(z; w)$: $\mathbb{F} \rightarrow \mathbb{X}$, $z \in \mathbb{F}$ indica o espaço latente, e $\mathrm{x} \in \mathbb{X}$ corresponde o espaço dos atributos, $w$ é um vetor que parametriza$f$. A FIGURA 13 mostra a visualização de uma rede neural autocodificadora.

FIGURA 13 – REDE NEURAL AUTOCODIFICADORA PADRÃO




FONTE: Adaptado de Canchumuni, Emerick e Pacheco (2019, p. 88).

A FIGURA 13 mostra que esse tipo de autocodificadora possui três elementos, o codificador, o código e o decodificador. O mapeamento entrada-saída é dado por $\mathrm{x}\rightarrow\tilde{\mathrm{x}}$, onde $\tilde{x}$ é um estimador da entrada $x$. Em termos gerais, $f(z;w)$ produz um operador identidade pelo estrangulamento dimensional de $x$, via espaço latente $z$. A estimativa é dada por $\tilde{x} = f_d(f_e(z;w_e);w_d)$, $f_d$ é o decodificador e $f_e$ é o codificador.

Le, Patterson e White (2018) argumentaram que, nesse estrangulamento, gera-se um tipo de generalização, inspirado no mapeamento por estruturas isomorfas bijetoras. No entanto, uma classe especial denominada de redes Autocodificadoras Variacionais, ou em inglês, *Variational autoencoders* (VAE) é mais robusta. Portanto, o VAE é uma função análoga a um mapeamento bem definido.

Para Canchumuni, Emerick e Pacheco (2019), o caráter bayesiano surge no momento da estimativa da distância entre as distribuições, $\mathrm{p(z|x)}\|\mathrm{p(x)}$. A formulação do problema de otimização fica definido como,

$$\mathcal{L}(x) = \mathcal{L}_{RE}(x) + \mathcal{D}_{KL}(p(z|x) \parallel p(z)) \tag{50}$$

em que $\mathcal{L}(\mathrm{x})$ é a função objetivo global, $\mathcal{L}_{RE}$ é a função objetivo do erro de reconstrução de $\mathrm{z}\rightarrow\mathrm{x}$, e $\mathcal{D}_{KL}$ é a métrica de divergência de Kullback-Leibler, associado a divergência das distribuições condicionais $\mathrm{p(z|x)}\|\mathrm{p(x)}$.

O termo $\mathcal{L}_{RE}$ é dado por

$$\mathcal{L}_{RE}(x) = -\frac{1}{N_x}\sum_{i=1}^{N_x}[x_i\, ln(\hat{x}_i) + (1-x_i)\, ln(1-\hat{x}_i)] \tag{51}$$

Em que se considera a função de entropia cruzada binária. Para essa função, as classes $k$ dos dados foram representados pela notação de codificação *one-hot* (ver seção 3.3.1). Já o termo $\mathcal{D}_{KL}$ é dado por

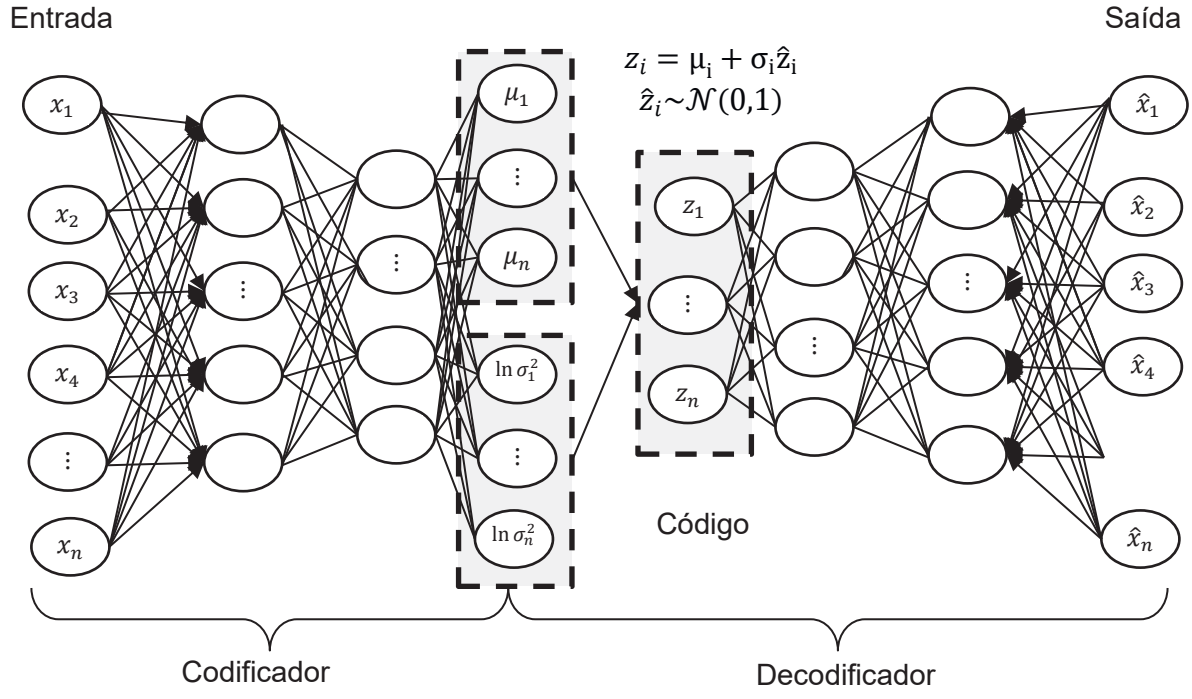
$$\mathcal{D}_{KL}(p(z|x) \parallel p(z)) = \frac{1}{2}\sum_{i=1}^{N_z}(\mu_i^2 + \sigma_i^2 - ln(\sigma_i^2) - 1) \tag{52}$$

em que $\mathcal{D}_{KL}$ é a divergência de Kullback-Leibler, enquanto $\mu_i$ e $\sigma_i$ são a média e o desvio padrão das amostras, respectivamente.

O cerne do VAE incide sobre a redução dimensional não linear do espaço dos atributos para $\mathrm{z}$ ao adicionar mais uma camada, que é responsável pela caracterização estatística da amostra $\mathrm{x}$. Portanto, a minimização é feita por método de gradiente estocástico bayesiano. A função objetivo deve possuir, no mínimo, a derivada de primeira ordem definida, ou seja, deve ser uma função de Classe 1.

O espaço dos atributos é denominado de espaço latente, dado por $z_i = \mu_\mathrm{i} + \sigma_\mathrm{i}\hat{z}_\mathrm{i}$ onde, $\hat{z}_i \sim \mathcal{N}(0,1)$, em que $\mathcal{N}$ representa uma função densidade de probabilidade gaussiana. A função $\mathcal{N}(0,1)$ possui média nula e desvio padrão unitário, em relação ao espaço amostral completo. A FIGURA 14 mostra esquematicamente a organização topológica de uma rede VAE.

FIGURA 14 – REDE NEURAL AUTOCODIFICADORA VARIACIONAL




FONTE: Adaptado de Canchumuni, Emerick e Pacheco (2019, p. 89).

Após o treinamento e ajuste do gradiente, a rede VAE produz uma representação reduzida de x (KINGMA; WELLING, 2013). Essa representação é costumeiramente interpretada como uma análise de componentes principais não

linear. Essa representação não linear permite a emulação prática de um algoritmo de agrupamento de atributos. Neste trabalho, o modelo VAE foi composto com a topologia mostrada na TABELA 4.

TABELA 4 – ARQUITETURA DA REDE AUTOCODIFICADORA VARIACIONAL (VAE)

| Camada | Dimensões | Número de parâmetros* | Função de ativação | Função de Perda |
|---|---|---|---|---|
| Entrada | (Nenhuma, 1600) | 0 | Nenhuma | Nenhuma |
| Densa (Codificador) | (Nenhuma, 64) | 102464 | ReLU | Nenhuma |
| Densa (Codificador) | (Nenhuma, 32) | 2080 | ReLU | Nenhuma |
| Densa (Mu) | (Nenhuma, 2) | 66 | Nenhuma | Nenhuma |
| Densa (Log_var) | (Nenhuma, 2) | 66 | Nenhuma | Nenhuma |
| Lambda (Amostragem) | (Nenhuma, 2) | 0 | Nenhuma | Nenhuma |
| Densa (Decodificador) | (Nenhuma, 32) | 96 | ReLU | Nenhuma |
| Densa (Decodificador) | (Nenhuma, 64) | 2112 | ReLU | Nenhuma |
| Densa (Saída) | (Nenhuma, 1600) | 104000 | Sigmoid | Entropia Cruzada Binária |
| Multiplicar | (Nenhuma,) | 0 | Nenhuma | Nenhuma |
| AddLoss | (Nenhuma,) | 0 | Nenhuma | Perda VAE |
| AddMetric | (Nenhuma,) | 0 | Nenhuma | Perda de Reconstrução |
| AddMetric_1 | (Nenhuma,) | 0 | Nenhuma | Perda KL |
| AddMetric_2 | (Nenhuma,) | 0 | Nenhuma | Perda VAE |

FONTE: O autor (2023).

NOTA: (*1) A quantidade de parâmetros foi calculada via biblioteca Keras, que é a biblioteca utilizada para treinamento do modelo. A palavra nenhuma, no contexto, (Nenhuma, X) representa uma dimensão extra, geralmente atribuída ao tamanho do lote, e X indicada a dimensionalidade da entrada.

Conforme exposto previamente, a rede variacional autocodificadora (VAE) é uma arquitetura de rede neural que é usada principalmente para realizar tarefas de redução de dimensionalidade e geração de dados semelhantes aos de entrada, conceito conhecido como inteligência artificial generativa (IA Generativa). Nessa tese, a dimensionalidade a ser comprimida foi representada pela variável z_dim, em que ‘z’ indica a representação codificada no espaço latente.

Em relação a configuração inicial a rede é inicializada com um tamanho de lote de 1 (treinamento online) e um número total de épocas de treinamento de 5. O

tamanho da camada oculta é definido como 64, e a dimensão do espaço latente, z_dim, é definida como 2.

A entrada é fornecida como dados de forma ( , 1600) para a camada de entrada. Duas camadas densas são empilhadas no codificador, cada uma com ativação, *Rectified Linear Unit* (ReLU). A primeira camada densa reduz a dimensionalidade dos dados para 64 unidades e, em seguida, a segunda camada densa reduz ainda mais para 32 unidades. Duas camadas densas independentes são usadas para calcular os parâmetros da distribuição latente, ou seja, a média $\mu_i$ e $\sigma_i$ e o logaritmo da variância $\sigma_i$.

Uma camada lambda é incorporada para amostrar pontos no espaço latente com base nas médias e log-variâncias calculadas anteriormente. Dito isso, é feito adicionando ruído gaussiano a partir de uma distribuição normal padronizada às médias multiplicadas pelas exponenciais das metades dos log-variâncias. O termo camada lambda refere-se ao funcional computacional que encapsula expressões numéricas ou matemáticas como parte de uma camada da rede neural.

Os pontos amostrados do espaço latente são então alimentados ao decodificador. O decodificador é composto por duas camadas densas com a função de ativação ReLU. A primeira camada densa aumenta a dimensionalidade dos dados de volta para 32 dimensões, e a segunda camada densa aumenta para 64 dimensões. A última camada densa tem uma ativação sigmoide e produz a saída, que deve se parecer com os dados de entrada originais (reconstrução).

Em relação à função de perda é uma combinação de duas partes, a saber: a perda de reconstrução e a perda de divergência KL. A perda de reconstrução é calculada usando a entropia cruzada binária entre a entrada original e a saída do decodificador.

A perda de divergência KL é calculada com base nas médias e log-variâncias da distribuição latente e é usada para regular o espaço latente. Isso posto, a perda total do VAE é a soma dessas duas perdas. O modelo VAE é compilado com o otimizador RMSPROP (TIELEMAN; HINTON, 2012). Durante o treinamento, o modelo é ajustado aos dados de treinamento por 5 épocas com um tamanho de lote de 1. A validação é realizada nos dados de teste.

Finalmente, o VAE é uma arquitetura que aprende a mapear dados de entrada para um espaço latente de dimensionalidade inferior, onde os pontos nesse espaço

podem ser amostrados e decodificados para gerar dados semelhantes aos de entrada. O modelo é treinado para minimizar a perda de reconstrução e ao mesmo tempo regular o espaço latente usando a perda de divergência KL.

Além do VAE, um modelo separado chamado codificador é construído para mapear dados de entrada para o espaço latente. Esse modelo é usado para codificar dados, o que é extensivamente explorado durante a proposta do algoritmo de transferência de aprendizagem. Devido a isso, essa capacidade do VAE é abordada na seção 3.6, em que será delineado um novo paradigma de aprendizagem.

## 3.6 ALGORITMO PROPOSTO

O algoritmo proposto neste trabalho apresenta uma abordagem híbrida entre a transferência de aprendizado baseado em projeção com o aprendizado baseado na métrica de distância de distribuição. O algoritmo é denominado de transferência de aprendizagem via Minimização Variacional Projetiva-Adaptativa não Paramétrica (MVPAnP).

O objetivo do MVPAnP é realizar uma busca de malha aleatória sobre o espaço latente gerado pelo VAE. O espaço latente resultante é obtido pela parametrização da regressão de componentes principais por núcleo, em inglês conhecido pelo acrônimo *kernel Principal Component Regression* (KPCR). A otimização da similaridade é expressa pela projeção no espaço latente da fonte em relação ao alvo, utilizando o produto interno. A minimização da similaridade é realizada implicitamente via o algoritmo Adam, conforme apresentando na seção 3.3.3 (CHOLLET et al., 2018).

O KPCR é uma extensão não linear do Método da Regressão sobre Componentes Principais, em inglês, *Principal Component Regression* (PCR) (JOLLIFFE, 1982). No método KPCR considera-se o problema padrão de regressão originário multivariado, transformando-o em um problema de autovetores não lineares por meio do KPC. Autovetores não lineares são vetores que não mantêm uma relação de proporcionalidade constante em transformações não lineares. Em seguida, deflaciona-se a matriz kernel centralizada no espaço dos atributos e obtém-se uma aproximação não linear. Dessa forma, com base na notação da seção 3.5.2, o KPCR fica definido como,

$$y = \Psi\xi + \epsilon \tag{53}$$

em que $y$ é um vetor com $N$ observações, $\Psi$ é a matriz de mapeamento no espaço dos atributos, $\xi$ é o vetor dos coeficientes de regressão, $\Psi$: $\mathbb{R}^p \rightarrow \mathcal{F}$, que é aplicada sobre $\Psi(x_i)$, onde $\{x_i\}_{i=1}^N$, e $\epsilon$ é o erro normal do estimador. Conforme o método KPCA mostrou-se que, ao utilizar o truque do kernel, soluciona-se o problema de autovalores generalizado da matriz de variância semiempírica, que é associada a $\Psi^T\Psi$. Com isso, projetam-se os autovalores, sobre o espaço dos atributos (ver Eq. 49) e obtém-se a estimativa da Eq. 53, como,

$$y = Bw + \epsilon \tag{54}$$

em que $B = \Psi V$, com dimensão de $n \times N$, sendo $V$ a matriz de autovetores não lineares, dada por

$$V = \sum_{i=1}^{n} \alpha_i \Psi(x_i) \tag{55}$$

Soluciona-se a regressão com base no estimador $\widehat{w}$, dado por

$$\widehat{w} = (B^T B)^{-1} B^T y \; = \; \Lambda^{-1} B^T y \tag{56}$$

em que $\Lambda$ é a matriz espectral deflacionada em relação à quantidade de componentes principais não lineares. O problema de autovalores advém das transformações $\lambda V = \hat{C}V \rightarrow k\tilde{u} = n\lambda\tilde{u} \rightarrow n\lambda\alpha = k\alpha$.

Assume-se que o espaço dos atributos esteja normalizado. Portanto, os autovalores são, $\lambda_k(\alpha_k\alpha_k) = 1$. Em seguida, a regressão fica definida como,

$$f(x, c) = \sum_{k=1}^{p} w_k \tilde{\lambda}_k^{\,-1/2} \sum_{i=1}^{n} \alpha_i^k K(x_i, x) + b \tag{57}$$

o que é equivalente a

$$f(x,c) = \sum_{i=1}^{n} c_i K(x_i, x) + b \tag{58}$$

onde, $c_i$ é

$$c_i = \left\{ \sum_{k=1}^{p} w_k \, \alpha_i^k \right\}_{i=1}^{n} \tag{59}$$

A matriz gaussiana foi aplicada como kernel para o aumento dimensional do espaço dos atributos, $\Psi$: $\mathbb{R}^p \rightarrow \mathcal{F}$, uma vez que, o núcleo gaussiano, dado pela Eq. 60, é similar ao codificador desenvolvido da rede VAE e essa classe de funções é utilizada nas redes neurais da base radial (LI et al, 2020).

$$K(i,j) = \Psi(x_i)^T \, \Psi\left(x_j\right) = e^{\left(\frac{\|x_i - x_j\|}{2\delta^2}\right)} \tag{60}$$

em que $K(i,j)$ é a matriz kernel, $\delta$ é um hiperparâmetro que informa a largura de banda da gaussiana, e $\|\cdot\|$ é a norma euclidiana. A Eq. 60. Relaciona-se com a Eq. 46, na medida que obtém a matriz $k$ via o emprego do truque kernel, ou seja, calcula-se o produto interno de $\Psi(x_i)$ e $\Psi\left(x_j\right)$ não diretamente no espaço $\Psi$, mas, no espaço vetorial, $x$ com base nas coordenadas $x_i$ e $x_j$. Isso é possível pois o produto interno no espaço de Hilbert é invariante a transformações vetoriais.

Logo que definida a hipótese do condicionamento dos dados centralizados no espaço dos atributos, para a solução do problema $n\lambda\alpha = k\alpha$, aplica-se a matriz kernel estimadora, $\widetilde{K}$, para garantir a homogeneidade entre os espaços matriciais, com

$$\widetilde{K} = K - 1_N K - K 1_N + 1_N K 1_N \tag{61}$$

em que é $\widetilde{K}$ é a matriz kernel centralizada no espaço dos atributos, $1_N$ é a matriz unitária quadrada com dimensão, $N \times N$, e $K$ é a matriz kernel, obtida numericamente pela expansão de $K(i,j)$ via Eq. 60.

No contexto da transferência de aprendizagem, a projeção de dados fora do espaço do estimador original $\{x_i\}_{i=1}^{N} \in \mathcal{D}$, no caso, não pertencentes ao domínio fonte, deve ser ajustada para que o estimador seja aplicado. Com isso, os dados do domínio alvo, $\{x_i^{\mathcal{T}}\}_{i=1}^{N} \in \mathcal{T}$ , são centralizados em relação $\mathcal{D}$ como

$$\widetilde{K}^{TESTE} = K^{TESTE} - 1_{N_t}K - K^{TESTE}1_N + 1_{N_t}K1_N \tag{62}$$

em que $\widetilde{K}^{TESTE}$ é a matriz kernel para os atributos fora do domínio fonte, $K$ é a matriz kernel, e $N_t$ é o número das amostras no domínio alvo.

Na literatura adota-se a notação de $\widetilde{K}^{TESTE}$ para evidenciar que os dados transformados são originários de uma amostra fora do domínio original (ROSIPAL; TREJO; CICHOCKI, 2000, ROSIPAL et al, 2001, WIBOWO; YAMAMOTO, 2012; YUAN; GE; SONG, 2014). Neste trabalho, alterou-se a nomenclatura de $K^{TESTE} \rightarrow \widetilde{K}^{\mathcal{T}}$, para indicar a integração dos dados amostrais provenientes do domínio alvo, $\mathcal{T}$.

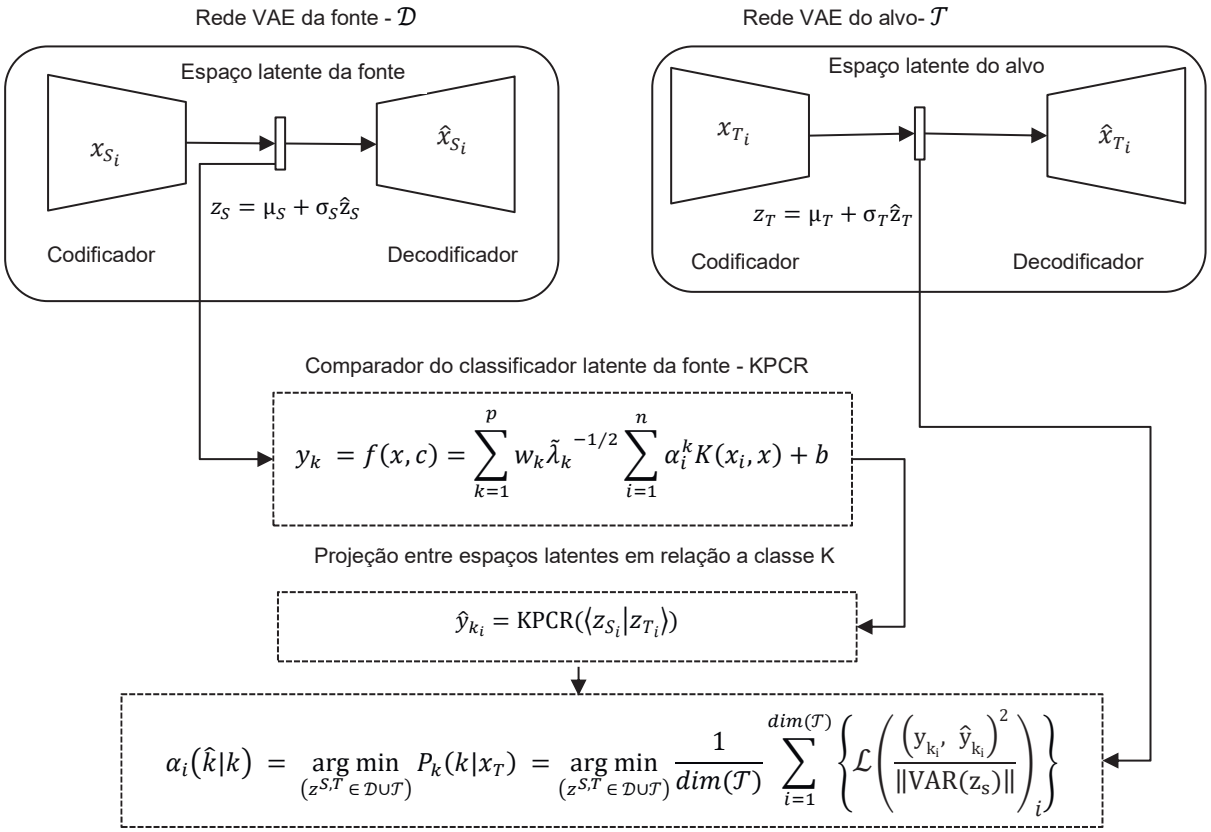
O objetivo do KPCR é extrapolar as classes na transferência de aprendizagem, com base no espaço latente da rede VAE, que proveu um agrupamento semi-supervisionado. Para tanto, as classes foram definidas conforme a notação da codificação *one-hot*, e centralizadas como

$$y_0 = \left(1_N - \frac{1}{N}1_N 1_N^T\right) y \tag{63}$$

em que $y_0$ é o vetor centralizado para as classes. A notação para as $k$ classes foi previamente definida na seção 3.3.1, dadas como $\{y_i\}_{i=1}^{k} \in \{0,1\}$.

Para o uso do KPCR, as classes foram expressas como $y = arg \max_k \{y_i\}_{i=1}^{k}$. A FIGURA 15 mostra o modelo esquemático do algoritmo MVPAnP. As entradas da rede autocodificadora variacional foram as respostas impulsivas das salas, com suas respectivas classes.

FIGURA 15 – FLUXOGRAMA DO ALGORITMO MVPAnP




FONTE: O autor (2023).

A solução do problema de minimização, mostrada na FIGURA 15, foi realizada pelo algoritmo Adam (CHOLLET et al., 2018). A escolha de usar o valor escalar do STI como variável preditora se deve ao cálculo padronizado do STI, conforme a norma IEC 60268-16 (IEC, 2011). Esse cálculo envolve a resposta impulsiva e a relação sinal-ruído. No entanto, também é possível introduzir diretamente a relação sinal-ruído no modelo, conforme demonstrado na Eq. 22. Ao garantir que a relação sinal-ruído seja menor que 15 dB, pode-se observar o impacto desse valor no STI.

Em termos de algoritmos, tem-se então um modelo esquemático da implementação computacional da função de perda MVPanP conforme o QUADRO 2.

QUADRO 2 – IMPLEMENTAÇÃO COMPUTACIONAL DA FUNÇÃO DE PERDA MVPAnP

| ETAPA | CÓDIGO |
|---|---|
| (1) | `from keras import backend as K`<br>`kl = tf.keras.losses.KLDivergence()` |
| (2) | `encoder_reconstruted = keras.models.load_model("VAE-encoder.model")`<br><br>`def KPCR_Prediction_Loss(input_tensor):`<br>`  z_latent = tensorflow_graph(np.expand_dims(input_tensor, axis=0),`<br>`  False, None).numpy()`<br>`  src_pred = kpcr.predict(z_latent)`<br><br>`  return src_pred`<br><br>`@tf.function(input_signature=[tf.TensorSpec(None, tf.float32)])`<br><br>`def KPCR_loss(input):`<br>`  y = tf.numpy_function(`<br>`  KPCR_Prediction_Loss,[input], tf.float32`<br>`  ).numpy()`<br>`  return y`<br><br>`def MVPAnP_Loss(data, y_pred):`<br>`  y_true = data[0][0]`<br>`  input2latent = data[0][1:1601]`<br>`  return K.mean(K.square(y_pred-y_true),axis=-1)`<br>`  - 0.1*kl(KPCR_loss(input2latent), y_true`<br>`  )` |
| (3) | `tf.config.run_functions_eagerly(True)` |
| (4) | `i = Input(shape=(1600,1))`<br>`x = Conv1D(filters=filter_num,`<br>`  kernel_size=kernel_size,`<br>`  strides=strides,`<br>`  kernel_initializer=kernel_initializer,`<br>`  activation=activation,`<br>`  input_shape=(1600, 1))(i)`<br>`x = Conv1D(100, 10, activation='relu')(x)`<br>`x = Conv1D(100, 10, activation='relu')(x)`<br>`x = MaxPooling1D(3)(x)`<br>`x = Conv1D(160, 10, activation='relu')(x)`<br>`x = Conv1D(160, 10, activation='relu')(x)`<br>`x = GlobalAveragePooling1D()(x)`<br>`x = Dropout(rate=dropout_rate)(x)`<br>`x = Dense(40, activation='relu')(x)`<br>`o = Dense(1, activation='sigmoid')(x)`<br>`model = Model(i, o)` |
| (5) | `model.compile(loss=MVPAnP_Loss, optimizer='adam')` |

FONTE: O autor (2023).

NOTAS: (1) – Define a importação do módulo do cálculo da divergência KL e importação do backend Keras; (2)-(3) Configuração para a execução do modelo de grafo do TensorFlow com o Keras como backend; e execução do grafo no modo *earger* para a compilação da função de perda MVPAnP e (4)–(5) Compilação do modelo 1D CONV NET com função custo customizada, MVPAnP_Loss.

Portanto, ao delinear o MVPAnP como um classificador não linear que se baseia na distância das projeções dos espaços latentes dos conjuntos de origem e alvo, como indicado,

$$\alpha_i(\hat{k}|k) = \underset{(z^{S,T} \in D \cup T)}{\arg\min}\, P_k(k|x_T) = \underset{(z^{S,T} \in D \cup T)}{\arg\min} \frac{1}{\dim(T)} \sum_{i=1}^{\dim(T)} \left\{ \mathcal{L}\left( \frac{\left(y_{k_i},\, \hat{y}_{k_i}\right)^2}{\|VAR(z_s)\|} \right)_i \right\} \tag{64}$$

da Eq. 64, têm-se os componentes de $\mathcal{L}(*)_i$ que são os equivalentes da função de perda $MVPAnP\_loss_i$, cujos componentes são

$$MSE\_loss_i = \frac{1}{n} \sum_{i=1}^{n} \left(y_{k_i} - \hat{y}_{k_i}\right)^2 \tag{65}$$

$$KPCR\_loss_i = \alpha_i(\hat{k}|k) = KPCR(\langle z_{S_i}(y_{k_i}) | z_{T_i}(\hat{y}_{k_i}) \rangle \tag{66}$$

$$KL_{loss_i} = 0.1 \sum_{i=1}^{n} KCPR\_loss_i \, log \left( \frac{KPCR\_loss_i}{y_{k_i}} \right)_i \tag{67}$$

$$MVPAnP\_loss_i = MSE\_loss_i - KL_{loss_i} \tag{68}$$

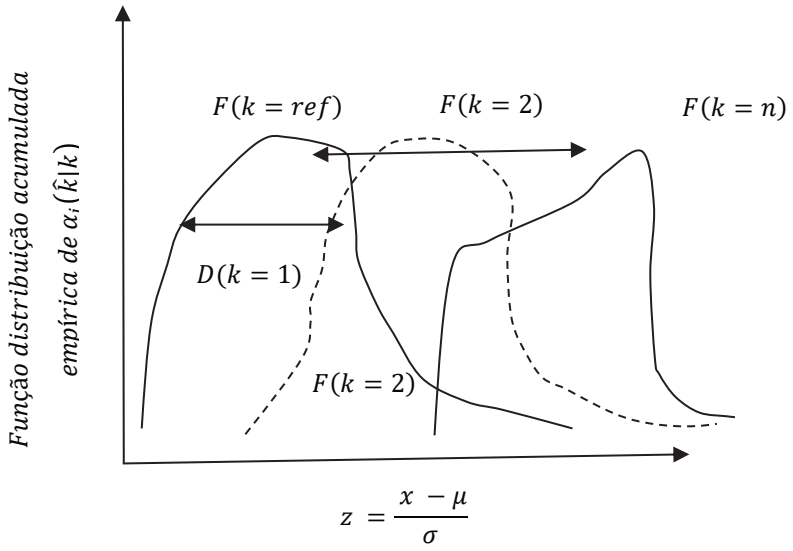
em que $\alpha_i(\hat{k}|k)$ é a estimativa das classes no espaço alvo, e o termo $\hat{y}_{k_i}$ é

$$\hat{y}_{k_i} = \tilde{f}(x) = \langle w^* | \psi(x) \rangle_{\mathcal{H}} = \langle \sum_{i=0}^{n} \left[ y \widehat{K}^T \right]_i \psi\left(x^{(i)}\right), \psi(x) \rangle_{\mathcal{H}} = y \widehat{K}^T \widehat{K}(X, x) \tag{69}$$

em que $\hat{y}_{k_i}$ é a estimativa gerada pelo KPCR, dada a entrada $x$, $\langle w^* | \psi(x) \rangle_{\mathcal{H}}$ é a representação de $\tilde{f}(x)$ consolidada no espaço de dimensionalidade expandida, realizada pelo truque Kernel, ao aplicar $\psi(x)$, dado um ajuste ótimo $w^*$, $\widehat{K}$ é a matriz kernel deflacionada em função ao número de autovalores, $X$ é o conjunto de entrada e $y$ é o vetor de classes conhecidas a priori para o ajuste de regressão.

A FIGURA 16 ilustra a interpretação geométrica do problema de otimização, o qual objetiva minimizar as diferenças entre as distribuições das classes via projeção dos espaços latentes. Essa abordagem permite considerar tanto o STI quanto a relação sinal-ruído no modelo, facilitando a análise do impacto da relação sinal-ruído no STI e contribuindo para a construção de um classificador robusto.

FIGURA 16 – INTERPRETAÇÃO GRÁFICA DO TESTE DE SIGNIFICÂNCIA K-S SOBRE O MVPAnP




FONTE: O autor (2023).

Assim, uma vez, obtidas as representações do espaço latente projetadas num espaço bidimensional, aplicou o teste de Kolmogorov-Smirnov, ou teste K-S. O teste K-S é uma ferramenta estatística amplamente utilizada para avaliar a aderência de uma distribuição de dados a uma distribuição teórica ou de referência. Desenvolvido pelos matemáticos Andrey Kolmogorov e Nikolai Smirnov em 1933, esse teste é particularmente útil para determinar se um conjunto de dados segue uma distribuição conhecida ou se difere significativamente dela (KOLMOGOROV, 1993; KOLMOGOROV; 1992). A estatística D do teste K-S, é o valor crítico que mede a máxima discrepância entre a função de distribuição acumulada (CDF) da amostra e a CDF da distribuição teórica. Essa discrepância é quantificada calculando-se a diferença absoluta máxima entre essas duas funções.

Neste trabalho, foi aplicada o *Kernel Density Metric Learning* (KDML) como função de distribuição empírica para a métrica de distância estatística para no espaço

das características, para obter a distribuição conhecida do teste K-S. Para a obtenção do KDML, foi utilizado o valor da estimativa da densidade de núcleo, KDE, que é mais conhecido pelo seu acrônimo em inglês, *Kernel Density Estimation* (KDE). O KDE é obtido por

$$\rho_K(y) = \sum_{i=1}^{N} K(y - x_i; h) \tag{70}$$

em que $\rho_K$ é a estimativa da densidade kernel para o ponto $y$, matematicamente, um kernel, $K(x; h)$, é uma função positiva que é controlada pelo parâmetro de largura de banda, $h$. Dada essa forma de kernel, estima-se de densidade em um ponto específico, $y$, dentro de um grupo de pontos, $x_i; i = 1, \cdots, N$. Neste trabalho foi adotado o Kernel gaussiano. Após a definição do KDE via Eq. 70, calculou-se o KDML conforme os procedimentos de He, Mao e Chen (2015).

Dessa forma, como corolário da formulação do MVPAnP, pode-se realizar o teste de significância estatística do grau de pertencimento das classes via o teste de Kolmogorov-Smirnov (ARNOLD; EMERSON, 2011). Abordagens similares existem, porém essas separam os algoritmos de agrupamento não supervisionado e os projetam sobre as classes usando o MMD (LONG et., 2013). Wang et al. (2020) propuseram um algoritmo similar usando o conceito de alinhamento no espaço dos atributos. Dessa forma, como critério de validação do MVPAnP, emprega-se o MMD.

## 3.7 VALIDAÇÃO EXPERIMENTAL DO ALGORITMO MVPAnP

Esta seção tem como objetivo descrever a validação experimental do modelo de previsão do STI, que aplicou redes neurais profundas convolucionais com o uso da transferência de aprendizagem via acoplamento do algoritmo MVPAnP como função custo. Além disso, visa também validar o novo algoritmo de transferência de aprendizagem, MVPAnP, que foi desenvolvido para melhorar o modelo de redes convolucionais em conjuntos de dados não apresentados durante o treinamento.

Para a validação do MVPAnP foram utilizadas medições *in situ*, do TR e STI, conforme mostra o QUADRO 3. Todas as medições foram realizadas em salas de

aula da Universidade Federal do Paraná, Campus Jardim das Américas/Centro Politécnico. As medições foram realizadas em salas vazias, sem a presença de alunos. Salienta-se que essas medições não foram apresentadas durante o treinamento das redes neurais profundas.

QUADRO 3 – MEDIÇÕES EXPERIMENTAIS DO TR E STI USADAS NA VALIDAÇÃO DO MVPAnP

| **SALA** | **QUANTIDADE DE PONTOS - STI** | **QUANTIDADE DE PONTOS - TR** | **REFERÊNCIA** |
|---|---|---|---|
| BIO ANF 14 | 25 | 5 | Nascimento (2019) |
| ENG QUI | 22 | 4 | Nascimento (2019) |
| PG 06 | 18 | 8 | Nascimento (2019) |
| ANF 04 | 12 | 5 | Nascimento (2019) |
| ANF 02 | 5 | 11 | Nascimento (2019) |
| EXATAS PA | 24 | 8 | Nascimento (2019) |
| EXATAS PA 05 | 26 | 11 | Nascimento (2019) |
| PG 04 | 40 | 5 | Herrmann (2018) |
| PG 15 | 32 | 12 | Herrmann (2018) |
| PG 03 | 19 | 5 | Herrmann (2018) |
| PG 05 LAAICA | 14 | 8 | Nascimento (2019) |
| PG 06 - V2 | 16 | 4 | Nascimento (2019) |
| PG 07 | 18 | 4 | Nascimento (2019) |
| TOTAL DE PONTOS | 90 | 301 | # |

FONTE: O autor (2023).

Portanto, a validação do algoritmo MVPAnP, aplicado na predição do STI, foi realizada com base num conjunto de (n = 301) medições de STI, num total de 13 salas. O tempo de reverberação foi medido em (n = 90) pontos. Com base nesse conjunto de medições de TR e ruído de fundo foram geradas uma amostra para a validação do algoritmo MVPAnP com 600 amostras.

O STI foi mensurado considerando a situação na qual o do ruído de fundo fosse contabilizado no momento das medições (SNR < 15 dB), para que assim, posteriormente, com variações do ruído de fundo, BGN, obtém-se novos valores do STI. Matematicamente esse procedimento foi delineado na Eq. 22, para obter os fatores de modulação.

Finalmente dado o valor da resposta impulsiva da excitação advinda do simulador de boca artificial passa por uma convolução com sinal do ruído de fundo advindo do banco de dados de ruído ambiental de Ko et. al (2017), apresentado na seção 3.2.1, o que aplicou o mesmo método de obtenção do STI para os dados sintéticos.

### 3.7.1 Medições *in situ*

As medições do STI foram realizadas seguindo as diretrizes da norma IEC 60268-16 (IEC, 2011), cujos procedimentos de cálculo foram previamente definidos na seção 2.4. O ruído de fundo foi medido por 5 minutos em salas de aula vazias com portas e janelas fechadas (para reduzir a interferência de ruídos externos), usando um medidor de nível de som B&K 2260.

O STI foi medido com os seguintes itens: *software* de acústica de sala DIRAC (B&K tipo 7841), versão 5.0, instalado em notebook Sony VAIO; placa de aquisição de dados Audio Interface ZE-0948; equalizador Behringer FBQ800; amplificador Gruppen LAB 300; simulador de boca B&K tipo 4227 e analisador de som em tempo real B&K 2260.

O espectro na saída do simulador da boca B&K tipo 4227 foi equalizado usando o sinal MLS com o filtro Pink + Blue, conforme recomendado pela IEC 60268-16 (IEC, 2011). Os ganhos no equalizador são ajustados para o nível de pressão sonora de referência de 60 dB, com um erro tolerável de ± 1 dB para as bandas de oitava de 125 Hz e 8 kHz.

O sinal com o espectro masculino foi usado para a calibração da fonte sonora (simulador de boca) e o receptor (analisador). A fonte e o receptor foram colocados um metro de distância, sendo o Leq ajustado para 60 dB(A).

Para as medições, o simulador da boca artificial foi colocado em uma única posição para representar o professor. O analisador B&K 2260 foi posicionado em 16 pontos homogeneamente espaçados nas salas de aula para representar os alunos. A FIGURA 17 mostra a disposição da instrumentação dentro de uma sala de aula.

FIGURA 17 – DISPOSIÇÃO EXPERIMENTAL PARA MEDIÇÕES DO STI E TR

FONTE: O autor (2023).

.

O tempo de reverberação foi mensurado de acordo com os procedimentos descritos na ISO 3382:2 (ISO, 2008), usando o Método de Resposta Impulsiva. O sinal de e-sweep gerado pelo software DIRAC 5.0 foi usado como excitação. Uma fonte dodecaédrica B&K 4296 foi posicionada a uma altura de 1,5 m do chão, e a mais de 1,2 m das paredes. No mínimo, cinco medições foram registradas com o receptor (analisador de som B&K 2260) em diferentes posições em cada sala de aula.

### 3.7.2 Métricas de validação: Rede 1D CONV NET com e sem TF

Conforme exposto na seção 3.3.1, adotou-se como a primeira entrada do modelo de redes neurais profundas, a densidade espectral da resposta impulsiva das salas PSD(RIR). A segunda entrada do modelo foi a densidade espectral do ruído de fundo medido nas salas, PSD(BGN). Após a definição das entradas, treinaram-se duas configurações de modelos, mas com a única diferença entre estes, sendo um modelo 1D CONV NET com função custo MSE e a outra com a função MSE + a

MVPAnP, conforme o exposto na TABELA 3. Em ambas as configurações foi aplicado o algoritmo ADAM como o otimizador.

Portanto, após o treino, a validação do algoritmo aplicado como função custo nas redes neurais profundas, 1D CONV NET para predições do STI foi realizada através das seguintes métricas: i) Erro Quadrático Médio, em inglês *Mean Squared Error* (MSE); Erro Médio Quadrático Percentual, em inglês, *Mean Squared Percentage Error* (MSPE) e Erro Quadrático Médio Percentual, em inglês, *Root Mean Squared Percentage Error* (RMSPE).

Essas métricas de erro foram avaliadas com o cálculo das diferenças entre os valores esperados e obtidos pela saída de cada modelo. Esses erros ponderam os valores de referência e os valores obtidos pelo modelo de redes neurais profundas para dados não apresentados durante o treinamento. Isso ocorre tanto com a abordagem de transferência de aprendizagem via MVPAnP quanto sem ela. O erro MSE mede a diferença média ao quadrado entre os valores previstos e reais,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{71}$$

por sua vez, a equação do MSPE, representa uma variação do MSE, que mede a diferença média ao quadrado em termos percentuais entre os valores previstos e reais,

$$MSPE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2 \times 100\% \tag{72}$$

e o RMSPE é a raiz quadrada do MSPE e fornece uma medida mais interpretável do erro médio percentual,

$$RMSPE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2 \times 100\%} \tag{73}$$

portanto, tem-se que RMSPE = √(MSPE). Os termos das Eqs. 71, 72 e 73, são: $n$ é o número de pontos de dados, $y_i$ é o valor real para o i-ésimo ponto de dados, $\hat{y}_i$ é o valor previsto para o i-ésimo ponto de dados.

Essas métricas de erro são úteis para avaliar o desempenho de modelos preditivos e compreender o quão bem eles se ajustam aos dados observados. Após a obtenção dos valores $y_i$ e $\hat{y}$, confeccionou-se a curva de calibração do modelo e calculou-se o coeficiente de correlação de Pearson dado por

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - y)^2}} \tag{74}$$

em que $\rho$ é o coeficiente de correlação de Pearson, $x_i$ e $y_i$ são os valores esperados e medidos respectivamente, com $i$ = até $n$, onde $n = 600$ são os números de amostras, $\bar{x}$ e $\bar{y}$ são as médias $x_i$ e $y_i$. Em relação a curva de calibração realizou-se uma análise estatística de resíduos a fim de verificar a homoscedasticidade e heterocedasticidade e do modelo preditivo do STI, 1D CONV NET, com e sem Transferência de Aprendizagem (TF).

# 4 RESULTADOS E DISCUSSÕES

Esta seção apresenta os resultados das medições *in-situ*, dos experimentos computacionais e das análises comparativas em relação à literatura corrente. Assim, num primeiro momento, sumariza-se brevemente a metodologia desenvolvida, seguindo-se da descrição dos resultados. Após, discutem-se as implicações desses resultados, e por conseguinte fornece-se uma interpretação de seu significado usando ferramentas de IA explicada (*eXplainable AI*). Por fim, comparam-se os resultados aqui determinados com trabalhos anteriores relevantes da literatura e destacam-se as diferenças e as semelhanças notáveis. Portanto, esta seção visa fornecer uma compreensão abrangente dos resultados da nova metodologia proposta e as suas implicações.

## 4.1 PROCESSAMENTO DOS DADOS

Conforme exposto na seção 3.2, dados sintéticos são os dados gerados artificialmente, em vez de coletados de observações do mundo real (*in locu*). Assim, no contexto da geração de respostas acústicas em salas de aula, dados sintéticos podem ser usados para simular as características sonoras pertinentes, como o tempo de reverberação, o nível de pressão sonora e o espectro de frequência da resposta ao impulso, para estudar os efeitos dessas características na inteligibilidade da fala e outros parâmetros acústicos.

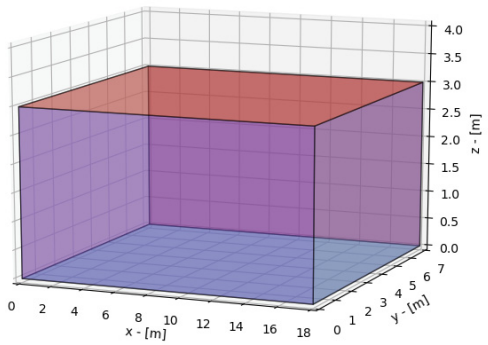
Os dados sintéticos podem ser particularmente úteis em situações em que não é prático ou viável coletar dados do mundo real, como ao estudar os efeitos de mudanças hipotéticas no projeto de uma sala de aula ou ao estudar ambientes acústicos específicos. Dados sintéticos também podem ser usados para avaliar e validar simulação acústica e ferramentas de predição, bem como para treinar modelos de aprendizado de máquina para tarefas como clonagem de voz e reconhecimento automático de fala.

### 4.1.1 Simulação acústica para a obtenção das salas virtuais

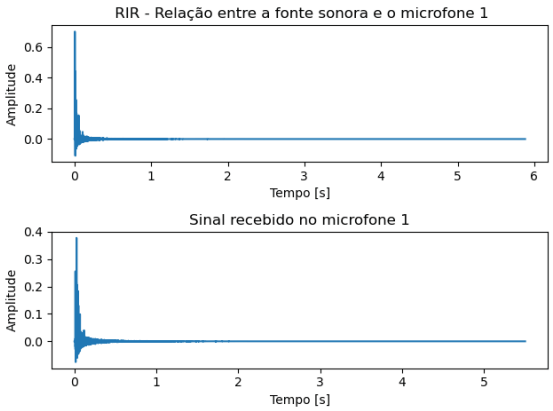
De acordo com a seção 3.2.1, o Método das Imagens foi usado para gerar 30000 salas virtuais com propriedades acústicas únicas. A FIGURA 18(a) ilustra um exemplo do modelo acústico de sala virtual e a respectiva resposta ao impulso da acústica da sala (RIR) correspondente, capturada na posição do receptor, denominado de microfone 1.

FIGURA 18 – MODELO VIRTUAL DAS SALAS DE AULA RETANGULARES

a) Sala virtual




b) Pressão sonora na posição do receptor




FONTE: O autor (2023).

LEGENDA: Simulação da resposta impulsiva dentro do ambiente virtual (a) Sala virtual gerada pelo Método das Imagens (ver seção 3.2.1), (b) Exemplo de captura da RIR e o respectivo sinal capturado numa posição aleatória dentro da sala.

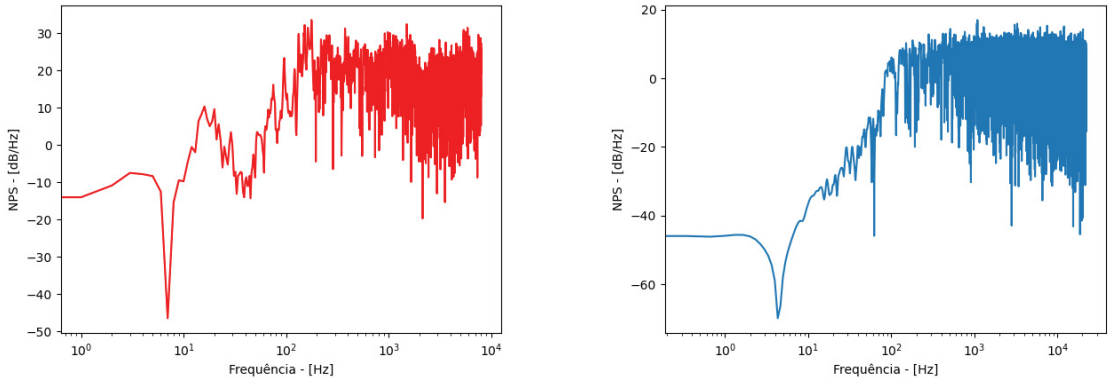
A FIGURA 18(b) apresenta o sinal da resposta impulsiva na sala virtual, para tanto, foi considerado um sinal de excitação impulsiva, com Leq de 60 dB, este sinal passou pela operação de convolução na sala virtual com o sinal de ruído de fundo que foi extraído de uma amostra de áudio do banco de dados de Ko et al. (2017). O sinal do ruído de fundo passou por filtros de banda de um terço de oitava e o Índice de Transmissão de Fala (STI) foi calculado usando as Eqs. 20 e 21.

Adicionalmente, a FIGURA 19 fornece uma representação gráfica do nível de pressão sonora da RIR nas amostras de áudio de medições *in situ* e dos dados sintéticos, assim como o resultado da aplicação do filtro de 1/3 de oitava nesses. Essa

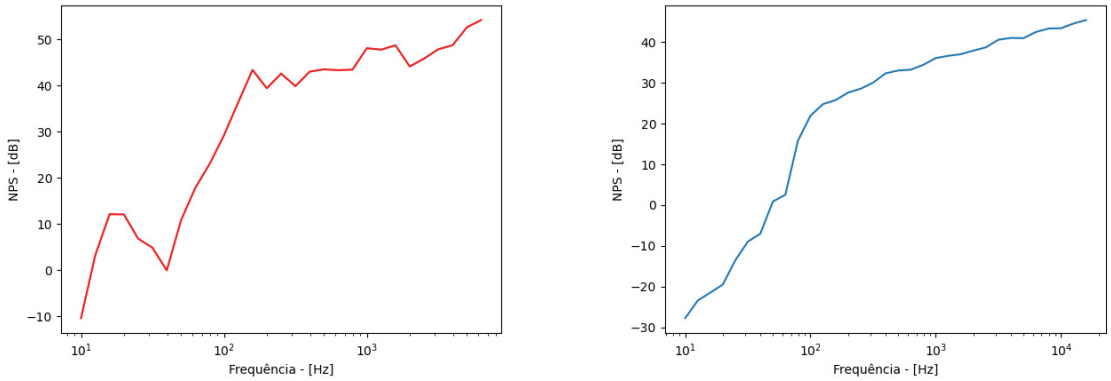
figura mostra, em particular, como os sinais sintéticos respondem ao filtro de 1/3 de oitava.

FIGURA 19 – COMPARAÇÃO DAS CURVAS RIR *IN SITU* VERSUS DADOS SINTÉTICOS

a) NPS (medição/*in situ*) b) NPS (dado sintético)




c) NPS em 1/3 de oitava (medição/*in situ*) d) NPS em 1/3 de oitava (dado sintético)




FONTE: O autor (2023).

LEGENDA: Comparação do Nível de Pressão Sonora (NPS) das medições da RIR *in situ versus* dados sintéticos obtidos de: (a) Sinal acústico de excitação RIR – medição *in situ*, (b) Sinal acústico de excitação RIR sintético em banda larga – simulado via o Método das Imagens, (c) Sinal acústico de excitação RIR – medição *in situ*, submetido ao filtro de banda de 1/3 de oitava e (d) Sinal acústico de excitação RIR sintético que foi submetido ao filtro de banda de 1/3 de oitava.

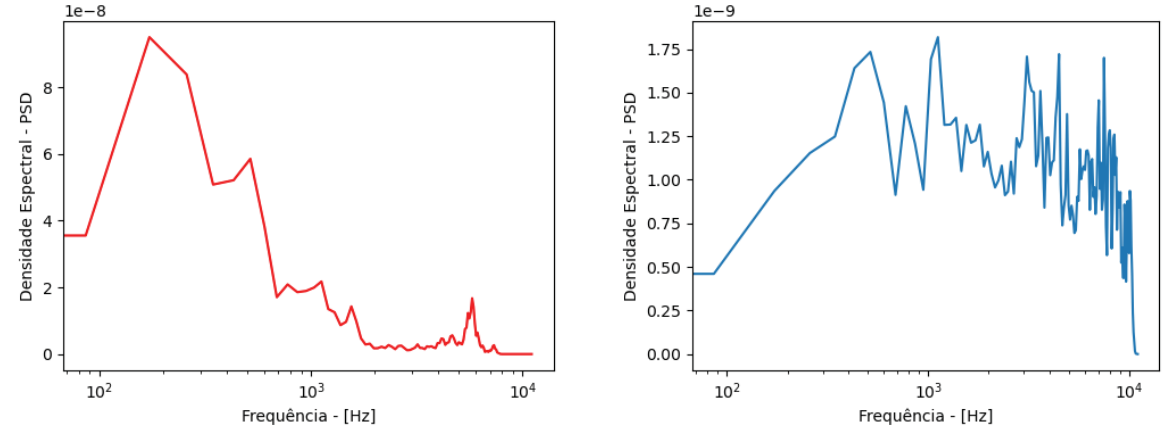
De posse das curvas RIR e do ruído de fundo de banda larga, conforme mostrado nas FIGURAS 18 e 19, respectivamente, e do correspondente valor de STI dessa combinação, fez-se a submissão desses como as entradas do modelo de aprendizagem profunda via a rede autocodificadora variacional (VAE), para a posterior realização do treinamento para os modelos, com ou sem transferência de aprendizagem.

Por sua vez, a FIGURA 20 exemplifica um par de entrada, composto de uma amostra do PSD do RIR e do PSD do sinal de ruído de fundo (BGN) do banco de dados de fonte medições *in situ* (conjunto de origem) e o conjunto de dados sintéticos (conjunto de alvo), respectivamente.

FIGURA 20 – DENSIDADE ESPECTRAL DE POTÊNCIA *IN SITU* VERSUS DADOS SINTÉTICOS

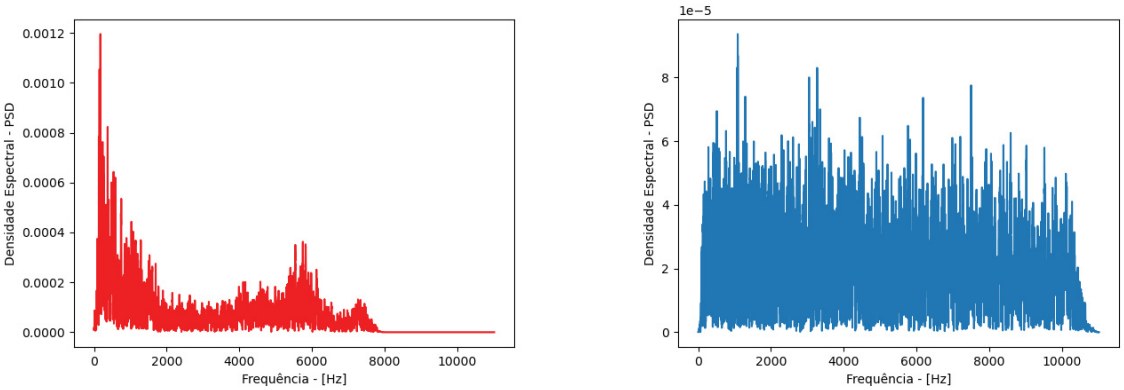
a) Densidade espectral (medição/in situ) - PSD(RIR)

b) Densidade espectral (dado sintético) PSD(RIR)




c) Densidade espectral (medição/*in situ*) PSD(BGN)

d) Densidade espectral (dado sintético) PSD(BGN)



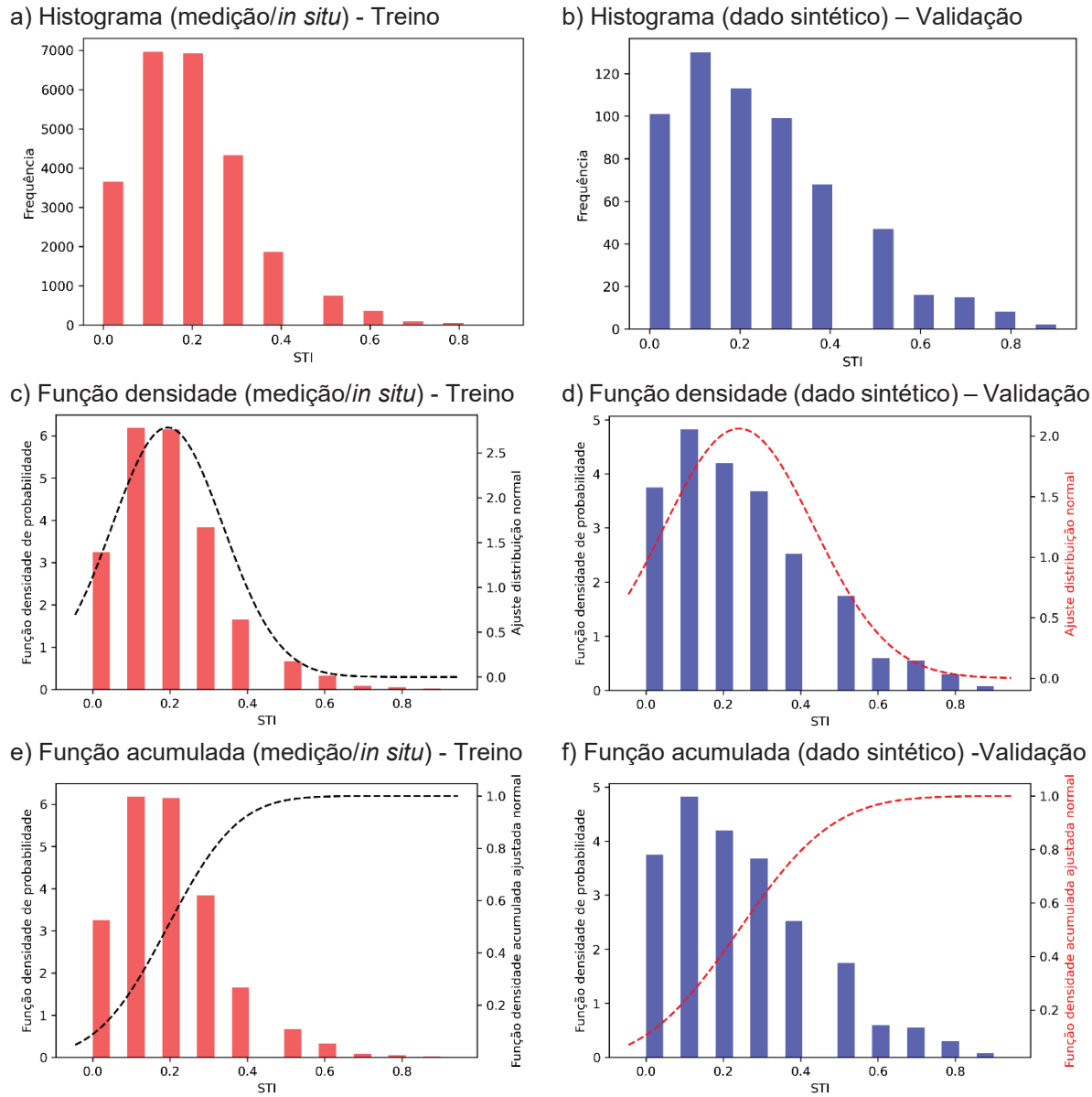

FONTE: O autor (2023).

LEGENDA: Exemplo do cálculo da densidade espectral (PSD) dos sinais da resposta ao impulso (RIR) e do ruído de fundo (BGN), , sendo; (a) PSD da RIR medido, (b) PSD do RIR sintético, (c) PSD do sinal de ruído de fundo medido e (d) PSD do sinal de ruído de fundo sintético.

Assim como na FIGURA 19, a disposição de energia do sinal sintético apresentou-se de forma menos uniforme do que o observado nas medidas *in situ*. A FIGURA 21 exibe o histograma dos valores de STI para as 25000 amostras, de treinamento do VAE, e para as 600 amostras para a validação do algoritmo MVPAnP,

em ambos os casos, os valores do STI foram caracterizados em 10 classes. Salienta-se que o termo amostra refere-se a um valor estimado de STI para sua respectiva sala virtual.

FIGURA 21 – HISTOGRAMA DO STI NOS CONJUNTOS ORIGEM E ALVO




FONTE: O autor (2023).

LEGENDA: Histograma das amostras de STI submetidas para treinamento em duas condições, sendo que, os gráficos à esquerda representam os dados de treinamento do VAE, enquanto os da direita são para a validação do algoritmo MVPAnP. As figuras do lado esquerdo pertencem ao domínio de origem, enquanto os gráficos da direita pertencem domínio de alvo. (a)-(b) Histograma, (c)-(d) Função Densidade de probabilidade (FDP) com ajustes para distribuição gaussiana (e)-(f) Função de Distribuição Acumulada (FDA).

De tal modo, a FIGURA 21 fornece a representação da distribuição dos valores de STI nos conjuntos de origem e alvo, permitindo a comparação com o intervalo esperado do STI em salas com características de volume similares. Sendo que (n = 25000) amostras foram utilizadas para o treino da rede neural VAE, e as outras 5000 amostras para validação do VAE. Cabe salientar que 600 medições foram utilizadas para validar o algoritmo MVPAnP. Para essa validação, foram treinadas duas redes neurais profundas convolucionais, uma com e a outra sem a função custo gerada via o algoritmo MVPAnP, conforme mostra o esquema da FIGURA 15 e a TABELA 3.

Ao observar a FIGURA 21(e)-(f), verifica-se que, cerca de 75% das amostras de STI foram menores que 0,5. Esse resultado está alinhado com os resultados determinados por Bistafa e Bradley (2000, p.68). Para eles, em salas de até 300 $m^3$ e com a relação ruído sinal, SNR (Ln – L1pm), de 15 dB e TR = 0,4 s resultaria no STI = 1,0, ou seja, 100% de inteligibilidade. Dadas essas condições, eles extrapolaram que para um TR acima de 0,80 s resultaria em STI < 0,5. Dessa forma, o TR médio calculado via as simulações foi de 1,3 s, o que extrapola o limite de 0,8 s e, como corolário, nessas condições o STI esperado é menor que 0,5.

### 4.1.2 Avaliação da dissimilaridade estatística entre os conjuntos de origem e alvo

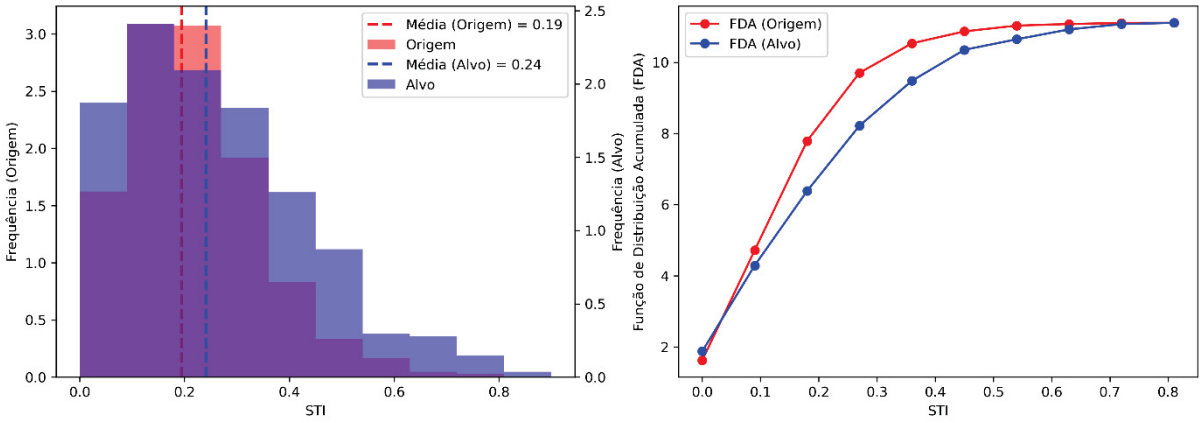
Sob a perspectiva da transferência de aprendizagem, foi estabelecido na seção 3.4 que há a necessidade de caracterizar a grau de semelhança entre os conjuntos de origem e alvo. Nesse contexto, aplicou-se o teste de Kolmogorov-Smirnov (K-S), com a hipótese nula (H0) postulando que não há diferença significativa entre os conjuntos de dados "Origem" e "Alvo". De tal modo, essencialmente, a hipótese nula aduz que esses conjuntos compartilham da mesma distribuição de probabilidade subjacente.

Na FIGURA 22 evidencia-se o resultado da aplicação do teste KS, pela comparação entre as Funções Distribuição Acumuladas (FDAs) não paramétricas de ambos os conjuntos com base somente na distribuição do STI. Ao inspecionar a FIGURA 22, foi determinada a rejeição da hipótese nula, (p-valor < 0,01), com esse p-valor excepcionalmente pequeno, aproximadamente 1,56e-09. Esse valor sugere que existe uma diferença estatística substancial entre os dois conjuntos de dados, que

vai além da variabilidade de amostragem aleatória. Essa distinção estatística mensurada possui ramificações cruciais quando vista a partir de uma perspectiva de transferência de aprendizagem.

FIGURA 22 – DISTRIBUIÇÕES DO STI NOS CONJUNTOS: ORIGEM *VERSUS* ALVO

a) Histograma do STI b) Função de Distribuição Acumulada




FONTE: O autor (2023).

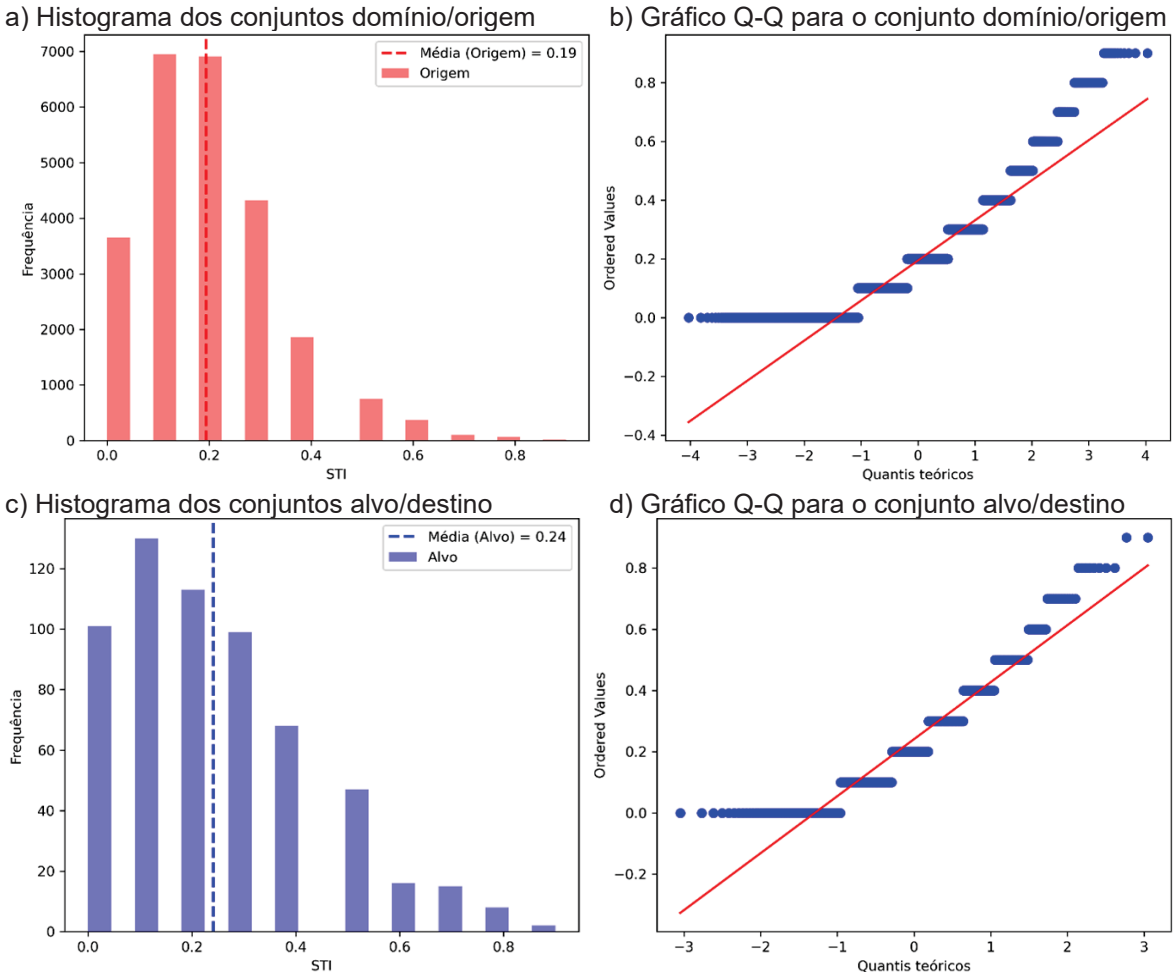
LEGENDA: Implementação do teste K-S sobre as distribuições do STI. (a) Histograma do STI para os conjuntos de origem e alvo; (b) Função de Distribuição Acumulada (FDA) dos valores do STI para os conjuntos de origem e alvo.

Dessa forma, os resultados indicam que, ao considerar apenas o STI como métrica para avaliar a similaridade entre as distribuições, observou-se uma diferença significativa entre os conjuntos de origem e alvo. No entanto, para a implementação da Transferência de Aprendizagem (TF), é necessário avaliar a métrica MMD, que é mais abrangente e não requer uma distribuição de referência. Além disso, nessa tese, foi tomada como referência o valor das distribuições geradas pelos espaços latentes como critérios para a aplicação da TF.

A FIGURA 23(a), tem-se o detalhamento em termos das distribuições do STI em ambos os conjuntos. Essa figura mostra o resultado do teste ANOVA, que indicou que que as distribuições possuem médias estatisticamente diferentes. De acordo com a FIGURA 23, é possível avaliar visualmente uma pequena similaridade estatística, na perspectiva gaussiana, entre os histogramas STI dos domínios origem (média = 0,19) e alvo (média = 0,24). Além disso, na FIGURA 23, é apresentado o teste de

normalidade de Shapiro, com Estatísticas-W de 0,91 e 0,92 para os conjuntos de origem e alvo, respectivamente.

FIGURA 23 – TESTE NORMALIDADE SOBRE OS DOMÍNIOS ORIGEM E ALVO




FONTE: O autor (2023).

LEGENDA: Implementação do teste de normalidade de Shapiro-Wilk sobre as distribuições do STI. O gráfico Q-Q demonstra o desvio da distribuição amostral do STI em relação à distribuição normal. (a)-(c) Histograma dos conjuntos alvo e domínio, (b)-(d) Gráfico Q-Q para os conjuntos alvo e domínio.

Esses valores 0,91 e 0,92 indicam que as distribuições do STI não podem ser consideradas como advindas de uma distribuição gaussiana, o que é constatado pelo desvio do ajuste linear no gráfico Q-Q. Portanto com a determinação de uma diferença significativa entre os conjuntos de dados de origem e alvo isso destaca a presença de uma mudança de distribuição entre esses domínios. Para navegar eficazmente por essa divergência via TF, técnicas de adaptação de domínio tornam-se imperativas,

tais com o emprego da métrica MMD sobre o espaço latente gerado via o treinamento da rede variacional autocodificadora VAE, que é demonstrado na seção 4.2.2.

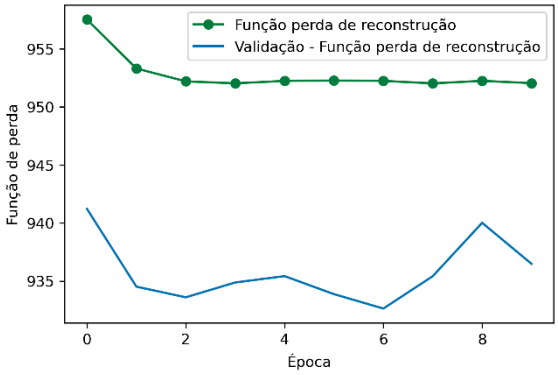
### 4.1.3 Resultados do treinamento da rede autocodificadora variacional - VAE

Depois que os conjuntos de dados de origem e alvo foram gerados, procedeu-se ao treinamento de uma rede neural profunda usando um modelo de rede autocodificadora variacional (VAE), ver TABELA 4, para obter o posterior aprendizado sobre o espaço latente (*embedding learning)*, para a aplicação da Transferência de Aprendizagem (TF).

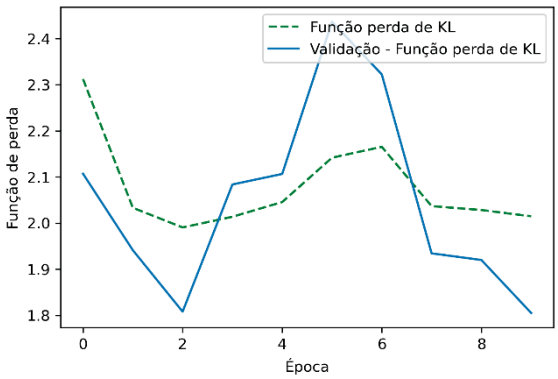
A FIGURA 24 exibe as métricas da função de perda global (soma dos componentes da função perda = reconstrução + divergência KL), conforme a Eq. 50, durante 10 épocas de treinamento da rede VAE.

FIGURA 24 – MÉTRICA DE RECONSTRUÇÃO DE PERDA PARA O TREINAMENTO DA REDE VAE

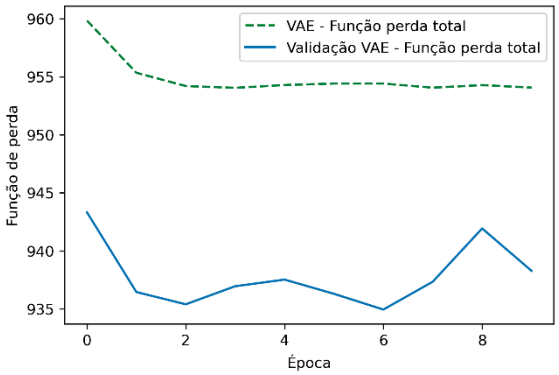
a) Função de perda – reconstrução




b) Função de perda – KL



c) Função de perda total




FONTE: O autor (2023).

LEGENDA: Decomposição dos valores da função perda - $\mathcal{L}$, Eq. 50, durante treinamento da rede variacional autocodificadora - VAE. A função perda total, contém dois componentes, a saber, o de reconstrução, $\mathcal{L}_{RE}$ , e o da divergência Kullback-Leibler (KL), $\mathcal{D}_{KL}$. Essas métricas foram calculadas para os conjuntos de treinamento, com 25000 amostras, e o conjunto de validação, com 5000 amostras, totalizando 30000 amostras/salas virtuais, sendo: (a) Época de treino *versus* função perda de reconstrução; (b) Época de treino *versus* função perda de divergência de Kullback-Leibler e (c) Época de treino *versus* função perda total.

A diferença entre os valores médios das curvas de treino e validação da rede variacional conforme são mostradas nas linhas azuis e verdes na FIGURA 24, deve-se ao efeito da dimensionalidade da quantidade amostral, (n = 25000 e n = 5000) dos respectivos conjuntos de treinamento e validação durante o cálculo da função de perda global.

A FIGURA 23(b) apresenta a função de perda de treinamento para todo o modelo, o que mostra a capacidade do modelo de prever com precisão os rótulos de classe dos dados. Fica claro, a partir desta figura, que o modelo foi calibrado, pois há, no treinamento, convergência para as funções de perda de reconstrução e de perda de divergência KL. Isso indica que o modelo pode generalizar para novos dados não submetidos ao treinamento. Assim, tem-se uma representação visual do desempenho do modelo no conjunto de dados de treinamento, indicando que ele pode aprender com precisão os padrões subjacentes nos dados devido à convergência.

Em termos de transferência de aprendizagem, esses resultados sugerem que o modelo VAE pode se adaptar efetivamente a novas tarefas por meio do ajuste fino dos conjuntos de dados relacionados estatisticamente. Tal procedimento foi realizado conforme exposto na seção 4.2.

## 4.2 OTIMIZAÇÃO DOS HIPERPARÂMETROS DA REDE AUTOCODIFICADORA VARIACIONAL

Neste trabalho, explorou-se a influência de diferentes parâmetros na precisão do codificador de saída (camada de saída da rede autocodificadora variacional). Os resultados indicam que os fatores mais cruciais que afetam os valores da função custo são as dimensões do espaço latente (z) e o número de neurônios em cada camada oculta da topologia do codificador-decodificador.

Adicionalmente, por meio de experimentos computacionais, demonstrou-se que a alteração desses parâmetros pode impactar significativamente a precisão do modelo. Esses resultados fornecem informações valiosas sobre a configuração ideal do modelo e destacam a importância da seleção cuidadosa de parâmetros para obter alta precisão para a generalização da rede neural.

### 4.2.1 Visualização dos *embeddings* para VAE

Ao realizar a validação cruzada em dez execuções (CV = 10) para cada combinação dos hiperparâmetros definidos na TABELA 3, verificou-se que a métrica de reconstrução permaneceu estável com um baixo desvio padrão. Esses resultados destacam a importância de selecionar os hiperparâmetros apropriados para alcançar o desempenho ideal para a geração de uma nova representação latente, dada as entradas PSD(RIR) e PSD(BGN). Na TABELA 5 consolidou as faixas de variação dos hiperparâmetros que foram ajustados durante a validação cruzada.

TABELA 5 – FAIXAS DE BUSCA PARA OTIMIZAÇÃO DE HIPERPARÂMETROS (HPO) DA REDE AUTOCODIFICADORA VARIACIONAL (VAE)

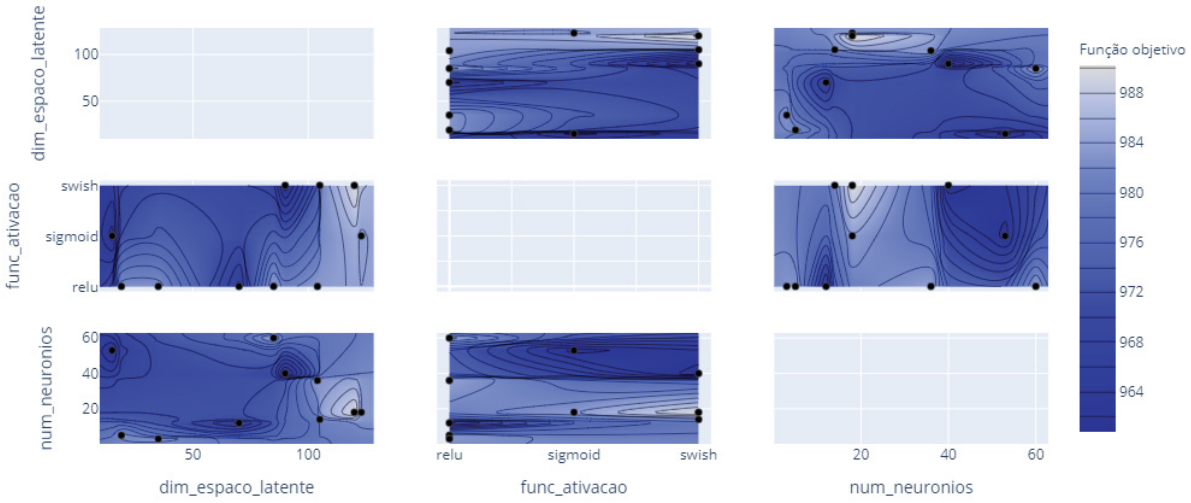
| **Nome da camada** | **Tipos dos parâmetros(*)** | **Faixa de variação do Hiperparâmetro** |
|---|---|---|
| Função de ativação | Nominal | relu, sigmoid e swish |
| Número de neurônios | Número inteiro | [2, 64] |
| Dimensão do espaço latente | Número inteiro | [2, 128] |

FONTE: O autor (2023).

NOTA: * (a) Nominal indica que a função de ativação é discreta e, portanto, só pode assumir algum dos valores elencados; (b) número inteiro, qualquer valor no intervalo dado.

Para avaliar visualmente o efeito de diferentes hiperparâmetros sobre a precisão do modelo VAE, examinaram-se combinações aleatórias da malha de busca desses parâmetros e calcularam-se os valores resultantes da função de custo. Os resultados apresentados na TABELA 4 são visualizados na FIGURA 25 como curvas de contorno.

FIGURA 25 – CURVAS DE CONTORNO VIA OTIMIZAÇÃO DOS HIPERPARÂMETROS DO VAE




FONTE: O autor (2023).

LEGENDA: Gráfico das curvas contorno para visualizar a otimização de hiperparâmetros do modelo de aprendizagem profunda VAE, descrito na TABELA 4. Sendo as variáveis, dim_espaco_latente - dimensão do espaço latente, func_ativacao - função de ativação e num_neuronios - número de neurônios.

Ao inspecionar o resultado da FIGURA 25, e com os valores da TABELA 6, observa-se que, aumentar o número de camadas ocultas e neurônios na primeira camada leva a uma melhoria na métrica da função de perda. Por outro lado, a alteração das funções de ativação não impactou de forma significativa o valor da função de perda. Desse modo, observou-se que o aumento da complexidade computacional via número de neurônios, ajudou a capturar padrões mais sutis nos dados e, por fim, melhorar o desempenho.

TABELA 6 – IMPORTÂNCIA RELATIVA DOS HIPERPARÂMETROS DA REDE VAE

| Hiperparâmetro avaliado | Importância sobre a Função objetivo(*) | Importância sobre o tempo de treinamento(*) |
|---|---|---|
| Função de ativação | 0,51 | 0,51 |
| Número de neurônios | 0,29 | 0,29 |
| Dimensão do espaço latente | 0,20 | 0,20 |

FONTE: O autor (2023).

NOTA: * (a) Esses resultados foram derivados dos algoritmos do módulo Optuna. (AKIBA et al., 2019)

Adicionalmente, verificou-se que o número de dimensões do espaço latente é um parâmetro crítico para o modelo VAE, pois determina o nível de compressão aplicado ao conjunto de dados. Para fins de visualização sobre a busca de malha, na TABELA 7, evidencia-se as combinações aleatórias aplicadas na otimização dos hiperparâmetros que foram derivados do módulo Optuna (AKIBA et al., 2019).

TABELA 7 – COMBINAÇÕES DOS HIPERPARÂMETROS NO TREINAMENTO DA REDE VAE

| Experimento | Duração Treinamento | Número de neurônios | Função de ativação | Dimensão espaço latente | Função objetivo |
|---|---|---|---|---|---|
| 1 | 00:08:27 | 106 | relu | 49 | 983,49 |
| 2 | 00:08:47 | 4 | swish | 51 | 992,61 |
| 3 | 00:08:53 | 41 | swish | 3 | 996,54 |
| 4 | 00:08:39 | 113 | relu | 18 | 965,72 |
| 5 | 00:08:48 | 28 | relu | 61 | 960,42 |
| 6 | 00:08:56 | 62 | swish | 56 | 966,15 |
| 7 | 00:08:50 | 125 | swish | 52 | 982,89 |
| 8 | 00:08:31 | 26 | relu | 4 | 983,08 |
| 9 | 00:08:39 | 69 | sigmoid | 6 | 983,04 |
| 10 | 00:08:37 | 11 | relu | 35 | 960,10* |

FONTE: O autor (2023).

NOTA: * (b) Essa foi a combinação de hiperparâmetros ótima. Com isso, “congelou-se” os pesos sinápticos do modelo VAE para o posterior uso do algoritmo apresentado na FIGURA 15.

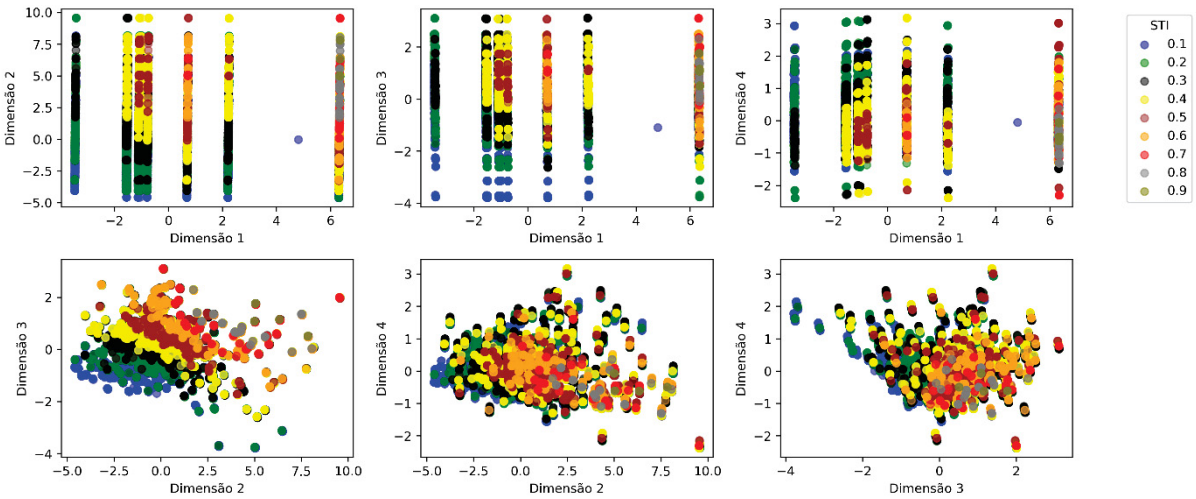
A TABELA 7 fornece evidências de que a otimização dos hiperparâmetros aumentou a precisão do modelo VAE, para o problema de reconstrução do codificador-decodificador, ao passo que determinou a configuração ótima da combinação dos hiperparâmetros dentro do espaço de busca durante o treinamento. Além disso, a TABELA 7 revela as demais combinações entre o número de neurônios e dimensão latente z, que foram determinadas por meio do processo de otimização de hiperparâmetros. Para tanto, deve-se observar a nota de rodapé da TABELA 7.

Ainda assim, nota-se que os resultados da TABELA 7 não são exaustivos, consolidou-se apenas com (CV = 10) rodadas para a otimização de hiperparâmetros. Estes dados constituem-se para visualização do processo de otimização dos hiperparâmetros. Além disso, os conceitos de IA explicável discutidos na seção 4.3 são utilizados para avaliar a interpretabilidade do espaço latente ótimo obtido. Essas descobertas sugerem que uma consideração cuidadosa deve ser dada ao número de dimensões ao usar um modelo VAE, e que, técnicas de IA explicáveis podem fornecer informações valiosas sobre o comportamento do modelo.

### 4.2.2 Estrutura topológica dos espaços latentes do VAE e cálculo da MMD

Depois de treinar o modelo VAE e selecionar a configuração ideal por meio da otimização de hiperparâmetros, ver TABELA 7, nota de rodapé (b), foi identificada a distribuição estatística para o conjunto treinamento e procedeu-se para a aplicação da transferência de aprendizagem profunda, para determinar o estado representacional de cada conjunto de dados de origem/alvo. Em seguida, aplicou-se o cálculo de perda (MMD) para avaliar o desempenho do modelo, nesses conjuntos de dados, de acordo com a FIGURA 26.

FIGURA 26 – DISPERSÃO PSD(RIR) E PSD(BGN) EM 4 COMPONENTES PRINCIPAIS




FONTE: O autor (2023).

LEGENDA: Projeção bidimensional do espaço das características (em 4 componentes principais) dos dados PSD(BGN) e PSD(RIR). Verifica-se que os dois principais componentes não foram capazes de abstrair uma representação definida de separabilidade dos agrupamentos.

O estado representacional do PCA é uma *proxy* da distribuição da projeção do conjunto de origem sobre um espaço bidimensional. É possível avaliar que o PCA não capturou, de modo suficiente, a variância dos dados, pois os agrupamentos gerados pelo PCA estão sobrepostos e mal separados, dificultando o discernimento de quaisquer padrões ou tendências claras nos dados. Na TABELA 8 consolidou-se esse resultado, nota-se que os dois componentes principais representaram mais de 80% de toda a variância explicada.

TABELA 8 – ANÁLISE DE VARIÂNCIA EXPLICADA SOBRE O ESPAÇO LATENTE VIA PCA

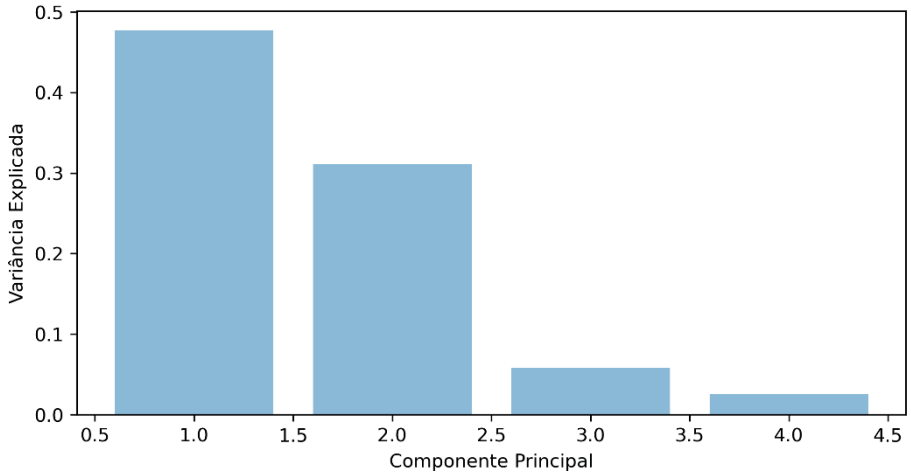
| Componente principal | Variância explicada | Variância cumulativa explicada |
|---|---|---|
| 1 | 0,50 | 0,50 |
| 2 | 0,30 | 0,80 |
| 3 | 0,10 | 0,90 |
| 4 | 0,00 | 1,00 |

FONTE: O autor (2023).

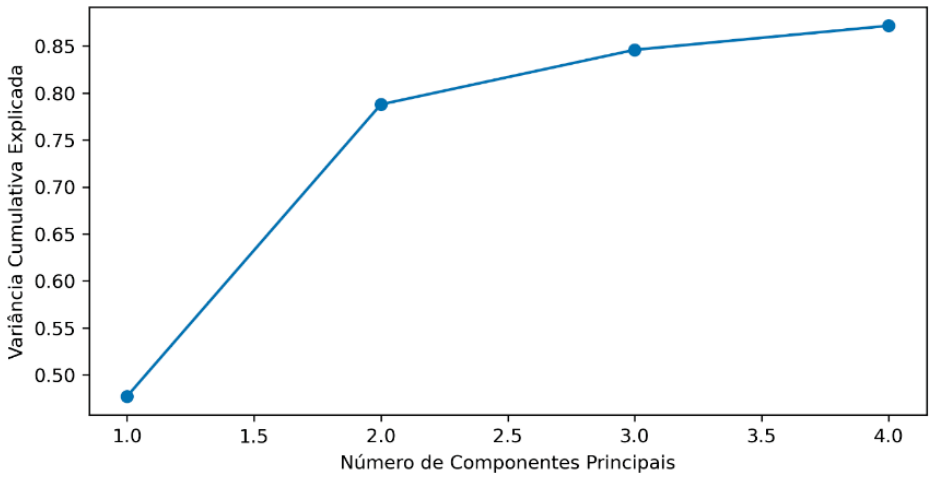
Na FIGURA 27 consolidou-se a representação gráfica dos resultados da análise de variância. Fica evidente que os dois principais componentes seriam suficientes para serem utilizados como variáveis de regressão em um modelo linear, tomando como critério apenas a condição de variância, porém ao observar a FIGURA 26 constatou-se que um modelo linear não é capaz de separar os agrupamentos emaranhados.

FIGURA 27 – CÁLCULO DE VARIÂNCIA DO STI VIA ANÁLISE DE COMPONENTE PRINCIPAIS

a) Variância explicada



b) Variância explicada acumulada




FONTE: O autor (2023).

LEGENDA: Análise de variância sobre os componentes principais lineares via PCA. (a) – Número de componentes principais *versus* variância explicada, (b) – Número de componentes principais *versus* variância acumulada.

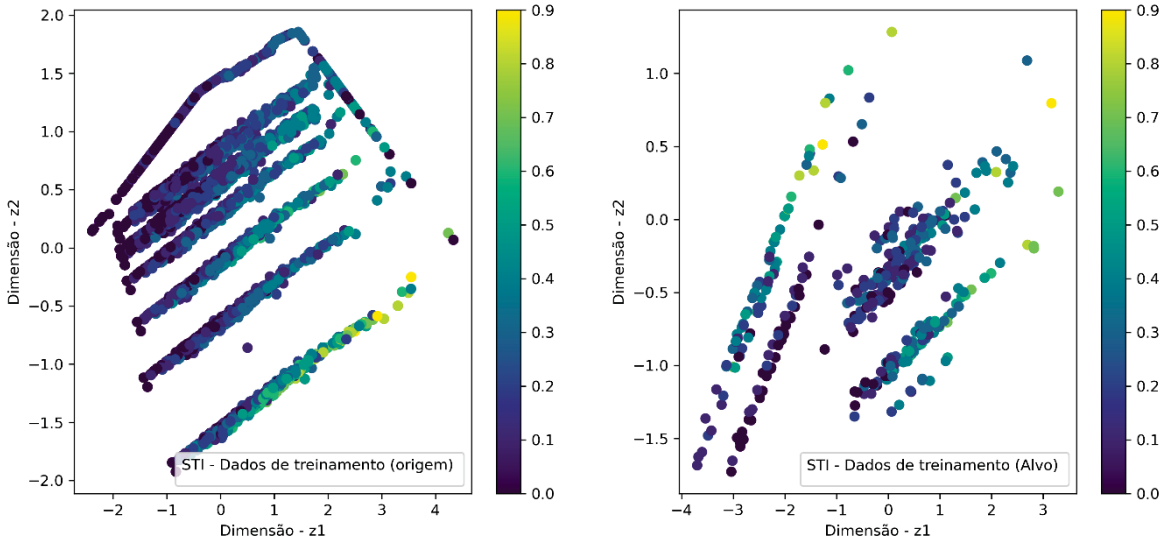
Como mencionado na seção 3.5, dois conjuntos de modelos foram usados neste estudo: (i) o conjunto de dados de treinamento/validação para a rede VAE, com 25000 e 5000 amostras respectivamente e o (ii) conjunto de dados de teste com 600 amostras que foram utilizados para treinar a rede convolucional profunda, nos cenários com e sem o algoritmo MVPAnP como a função custo. A arquitetura autocodificadora foi, então, utilizada para construir uma representação simplificada das classes no espaço latente, que foi projetada em um espaço bidimensional.

Nessa mesma linha de análise, porém abarcando a representação embutida via espaço latente, (z1; z2) que foi obtido após o treinamento da VAE, tem-se na FIGURA 28 apresenta os valores das classes atribuídas ao espaço latente, com uma projeção nas duas primeiras dimensões usando o VAE. Esta figura fornece uma representação da relação entre as classes do STI e sua representação no espaço latente, permitindo uma compreensão da dispersão após modelo gerar um novo espaço das características (*feature space*) para o conjunto alvo, baseado somente no conjunto de treinamento do VAE.

FIGURA 28 – PROJEÇÃO DO ESPAÇO LATENTE GERADO PELA REDE VAE

a) Espaço latente do conjunto de domínio – treinamento

b) Espaço latente do conjunto de alvo - treinamento




FONTE: O autor (2023).

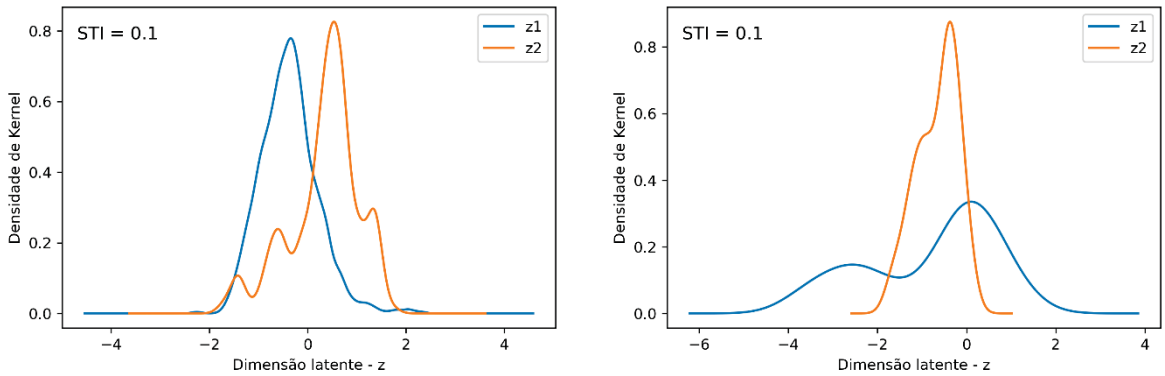
LEGENDA: (a)-(b) Gráficos da projeção bidimensional do espaço latente: z1 *versus* z2 - (componentes principais não lineares). Verifica-se que os dois principais componentes foram capazes de abstrair uma representação definida para separabilidade dos agrupamentos correspondentes aos valores STI com base no treinamento da rede VAE.

A FIGURA 29(a) mostra um exemplo, para as classes (k = 0,1; 0,3; 0,6 e 0,9 do STI) para o conjunto de dados de treinamento, enquanto a FIGURA 29(b) mostra a mesma classe, mas nos dados não vistos durante o treinamento da rede VAE, ou seja, foi apresentado um novo conjunto de testes. Essas figuras fornecem visualizações da capacidade do modelo de representar as classes no espaço latente e de generalizar para dados apresentados durante o treinamento.

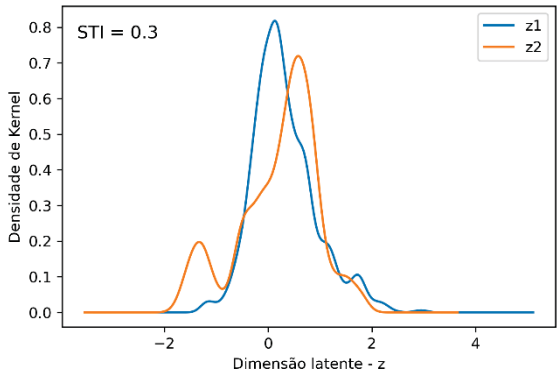
FIGURA 29 – COORDENADAS LATENTES *VERSUS* DENSIDADE DE NÚCLEO VIA KDE

a) KDE 1D (z1 e z2) – STI [0, 0,1[ - Domínio

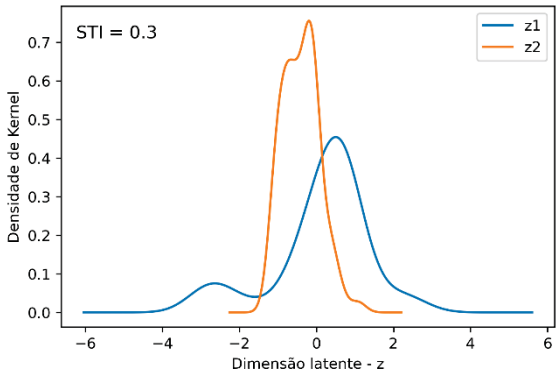
b) KDE 1D - (z1 e z2) – STI [0, 0,1[ – Alvo



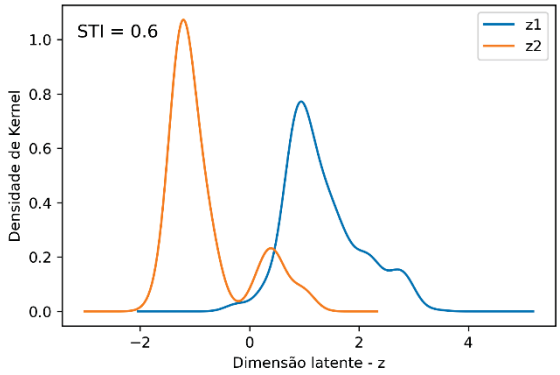
c) KDE 1D (z1 e z2) – STI [0.2,0.3[ - Domínio



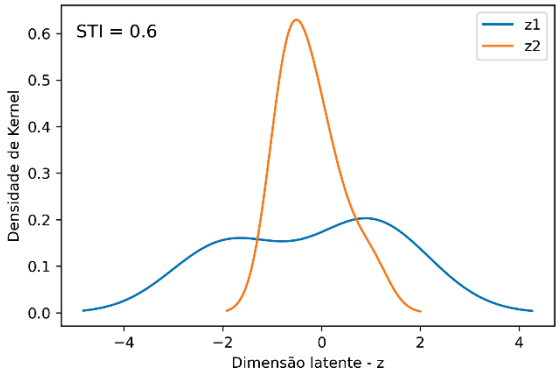

d) KDE 1D - (z1 e z2) – STI [0.2,0.3[ – Alvo



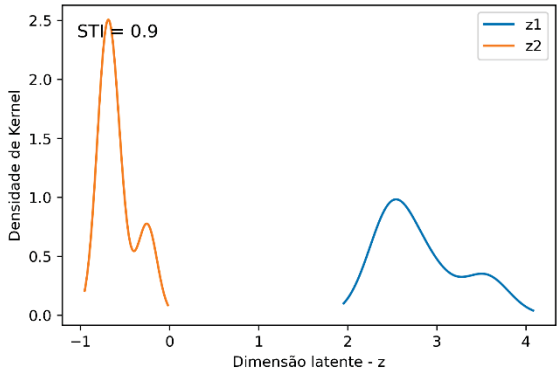

e) KDE 1D (z1 e z2) – STI [0.5,0.6[ - Domínio



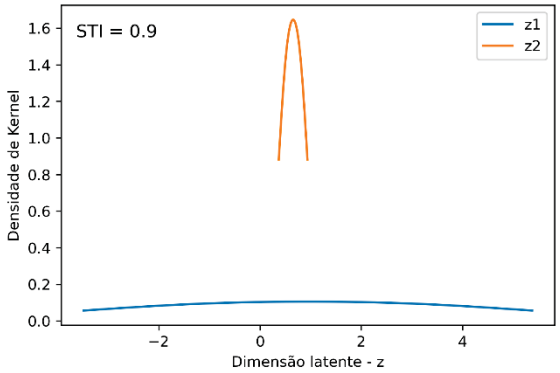

f) KDE 1D - (z1 e z2) – STI [0.5,0.6[ – Alvo




g) KDE 1D (z1 e z2) – STI [0.8,0.9[ - Domínio




h) KDE 1D - (z1 e z2) – STI [0.8,0.9[ – Alvo




FONTE: O autor (2023).

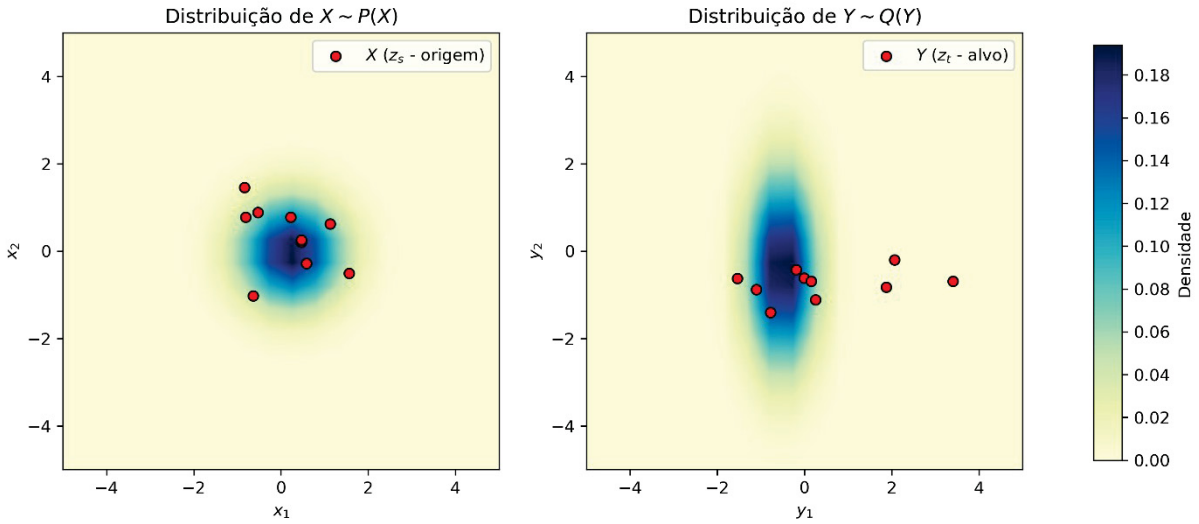
LEGENDA: Representação da densidade KDE calculada para os domínios de alvo e origem. Nestes gráficos os dados foram estratificados em função das coordenadas do espaço latente (z1, z2) por meio do cálculo KDE via Eq. 66. O algoritmo do modelo VAE pertence a classe de IA generativa, e, portanto, os espaços latentes gerados em cada treinamento serão notadamente díspares. Mas, a representação da posição relativa de cada uma dessas curvas produz uma localização inequívoca do STI.

Ao comparar as coordenadas latentes z1 e z2 por meio da análise de *Kernel Density Estimation* (KDE), é evidente, a partir das FIGURAS 27(a) e 27(b), que existe uma notável concordância na posição relativa dessas coordenadas. Isso ocorre

devido à posição relativa das coordenadas latentes, que demonstra uma alta concordância, sendo um fator crucial para a precisão da predição.

Para validar quantitativamente a similaridade acima declarada, calculou-se a métrica MMD. O valor da métrica MMD foi de 0,53, para esses resultados considera-se a diferença entre as distribuições do conjunto de origem e alvo via Eq. 43.

FIGURA 30 – MÉTRICA MMD PARA DOMÍNIO DE ORIGEM E ALVO




FONTE: O autor (2023).

LEGENDA: Avaliação da métrica MMD no espaço latente z1 e z2 gerado pelo treinamento da rede autocodificadora variacional (VAE). Nesse os pontos vermelhos representam a localização média do STI para cada um dos valores 0, 0,1, 0,2 ... 0,9.

A métrica MMD é uma medida de diferença distributiva não paramétrica, ou seja, não assume nenhuma forma funcional para as distribuições subjacentes. Sua compreensão depende da dimensionalidade das variáveis de entrada, e o resíduo das diferenças entre as projeções são acumuladas e ponderadas pela dimensão do espaço avaliado. Ao contrário do teste K-S, este considerou apenas a diferença residual da Função de Distribuição Acumulada (FDA) de cada conjunto e estabeleceu-se o teste de verificação via a estatística D.

A TABELA 8 mostra o teste de Kolmogorov–Smirnov para medir as funções de distribuição do KDE, que usaram a estatística D para testar a hipótese nula da distribuição de espaço latente do KDE derivada da mesma amostra distributiva.

TABELA 9 – TESTE DE KOLMOGOROV-SMIRNOV NOS ESPAÇOS LATENTE DA REDE VAE

| | | | | | **STI** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **STI** | 0,0 | 0,1 | 0,2 | 0,3 | 0,4 | 0,5 | 0,6 | 0,7 | 0,8 | 0,9 |
| 0,0 | 0,094* | 0,148* | 0,203 | 0,383 | 0,359 | 0,242 | 0,18 | 0,117* | 0,109* | 0,094* |
| 0,1 | 0 | 0,094* | 0,164* | 0,367 | 0,336 | 0,227 | 0,141* | 0,086* | 0,125* | 0 |
| 0,2 | 0 | 0 | 0,125* | 0,312 | 0,289 | 0,188 | 0,102* | 0,133* | 0,172 | 0 |
| 0,3 | 0 | 0 | 0 | 0,297 | 0,266 | 0,156* | 0,133* | 0,219 | 0,242 | 0 |
| 0,4 | 0 | 0 | 0 | 0 | 0,219 | 0,336 | 0,32 | 0,367 | 0,414 | 0 |
| 0,5 | 0 | 0 | 0 | 0 | 0 | 0,234 | 0,281 | 0,32 | 0,383 | 0 |
| 0,6 | 0 | 0 | 0 | 0 | 0 | 0 | 0,141* | 0,211 | 0,281 | 0 |
| 0,7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,133* | 0,195 | 0 |
| 0,8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,117* | 0 |
| 0,9 | 0,094* | 0,148* | 0,203 | 0,383 | 0,359 | 0,242 | 0,18 | 0,117* | 0,109* | 0,094* |

FONTE: O autor (2023).

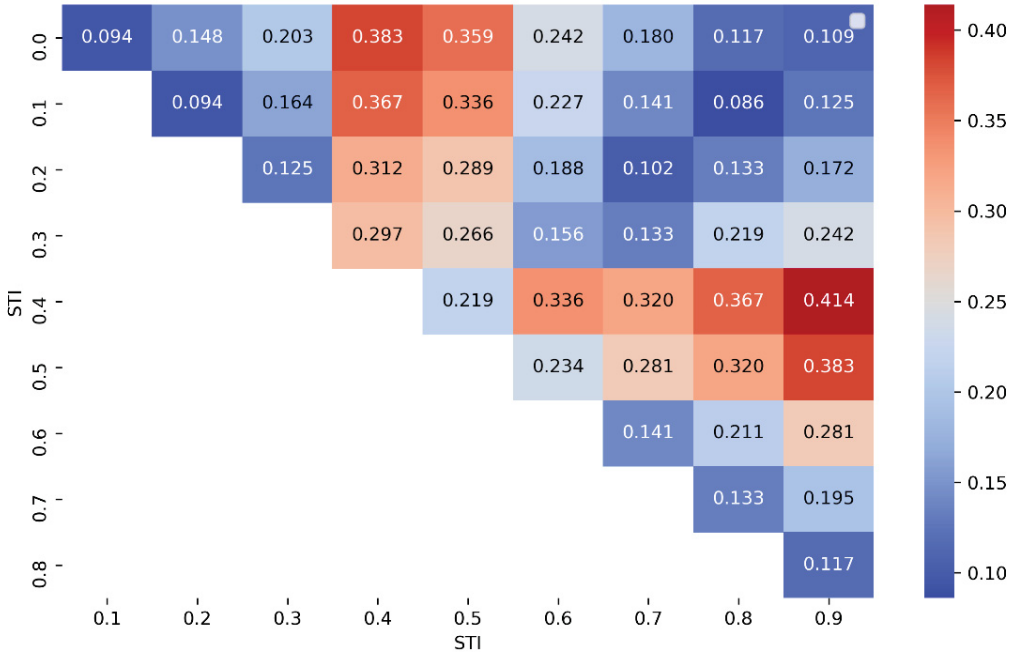
NOTA: * indica que a hipótese nula foi aceita com p-valor > 0,05. Os valores apresentados são a Estatística-D do teste Kolmogorov-Smirnov.

Conforme a TABELA 9, as estatísticas D destacaram grande concordância ao identificar que os cruzamentos das distribuições no espaço latente advêm de distribuições similares. Esses resultados demonstram a eficácia da métrica MMD como métrica de distância-distribuição para o emprego da transferência de aprendizagem.

Cabe salientar que a dimensionalidade do espaço latente do STI foi derivada de 25000 amostras, normalizadas entre 0 e 1, isso a torna robusta para diferentes tipos de distribuições estatísticas e permite que seja aplicado a uma ampla gama de conjuntos de dados, como os do STI. Ademais, com essa validação semiempírica, é possível aplicar a transferência de aprendizagem (TF), pois a hipótese de similaridade está em conformidade com a distribuição estatística do STI em ambos os conjuntos de domínio e alvo (ver FIGURA 21).

Além disso, notou-se que apesar do acentuado alinhamento, algumas das distribuições não são estatisticamente equivalentes para certas faixas de STI. Por fim, é necessário estabelecer expectativas realistas para o desempenho do modelo durante a transferência de aprendizagem, ponderando, dessa forma, as intrincadas diferenças de domínio reveladas pelo teste K-S, o que pode ser constatado na FIGURA 31.

FIGURA 31 – MAPA DE CALOR DO TESTE KS SOBRE OS VALORES DE KDE




FONTE: O autor (2023).

NOTA: Avaliação amostral da métrica KS, sobre cada possível combinação dois a dois das curvas KDE. Verifica-se acentuada concordância entre as distribuições, informando que os resultados do VAE indicam que a distribuição determinada para o espaço (z1, z2) deriva de uma mesma distribuição. Salienta-se que a métrica KS não informa qual é a distribuição, apenas as compara.

Em geral, a diferença observada nos resultados dos testes KS, destaca a importância de mensurar as disparidades entre os domínios de maneira sistemática e assim então empregar as estratégias de adaptação de domínio pertinentes, para facilitar a transferência de aprendizagem de forma robusta e eficaz. E por conseguinte, ao findar o treinamento do modelo VAE foram obtidas as respectivas coordenadas latentes (z1, z2) para ambos os conjuntos de origem e alvo.

## 4.3 IA EXPLICÁVEL APLICADA AO ESPAÇO LATENTE

Quando são comparadas as coordenadas latentes z1 e z2 do MMD, nas FIGURA 29(a) e 29(b) estas revelam um nível elevado de concordância, com base, no MMD (FIGURA 30) e nos valores do teste K-S, conforme mostra a TABELA 9. No entanto, para obter uma compreensão mais profunda do que está acontecendo dentro

do modelo VAE, em termos representacionais, é essencial fornecer explicabilidade ao espaço latente gerado.

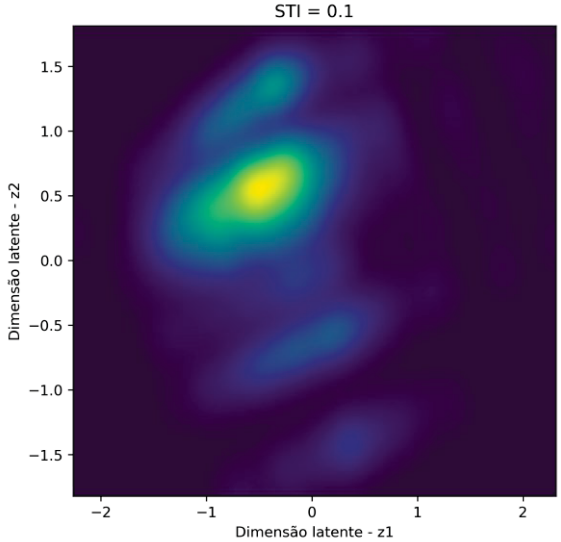
Isso posto, neste trabalho, foram avaliadas duas abordagens aplicadas para atingir esse objetivo, quais sejam: (i) a otimização de hiperparâmetros – conforme a seção 4.2; (ii) análise de sensibilidade da distribuição do espaço latente, conforme mostrado na FIGURA 29. Essas abordagens permitem uma compreensão mais completa do comportamento do modelo e melhoram sua interpretabilidade.

As descobertas derivadas dos itens (i) e (ii) fornecem informações valiosas sobre os fatores que influenciam a precisão do modelo e dão suporte ao uso da abordagem de TF. Pois, uma vez que, a minimização da discrepância média máxima (MMD) é um método usado na transferência de aprendizagem para alinhar as distribuições estatísticas dos domínios de alvo e de origem. Ela faz isso, ao minimizar a distância entre as duas distribuições. Uma maneira de visualizar o alinhamento das distribuições usando a minimização MMD é usar um gráfico de estimativa de densidade de kernel 2D (KDE), conforme mostrado na FIGURA 32.

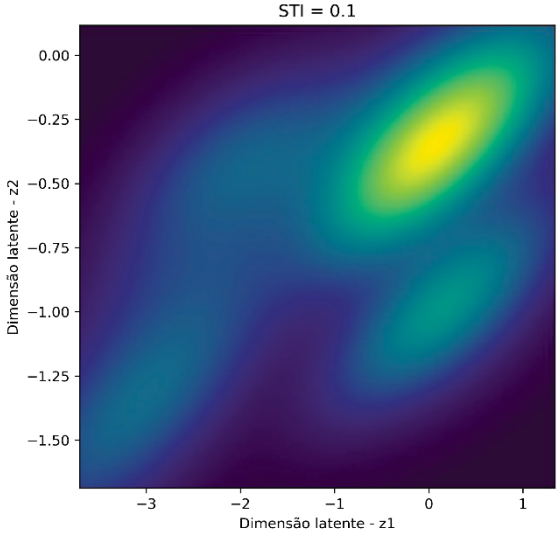
Um gráfico do KDE bidimensional é uma forma não paramétrica de estimar a função de densidade de probabilidade de uma variável aleatória e pode ser usado para visualizar a forma de uma distribuição. O uso de um gráfico do KDE fornece uma representação visual do alinhamento das distribuições e da eficácia do processo de minimização da métrica MMD.

FIGURA 32 – COORDENADAS LATENTES DA DISTRIBUIÇÃO BIDIMENSIONAL - KDE

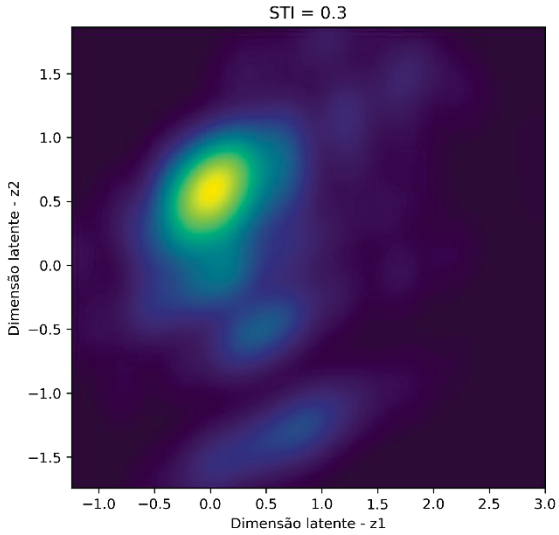
a) KDE 1D (z1 e z2) – STI [0, 0.1[ - Domínio



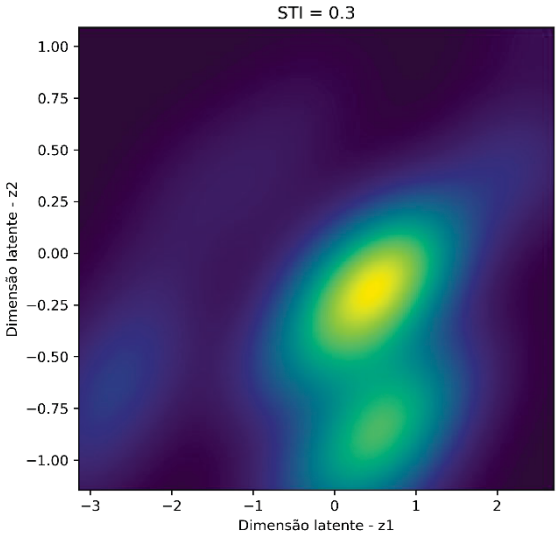

b) KDE 1D - (z1 e z2) – STI [0, 0.1[ – Alvo



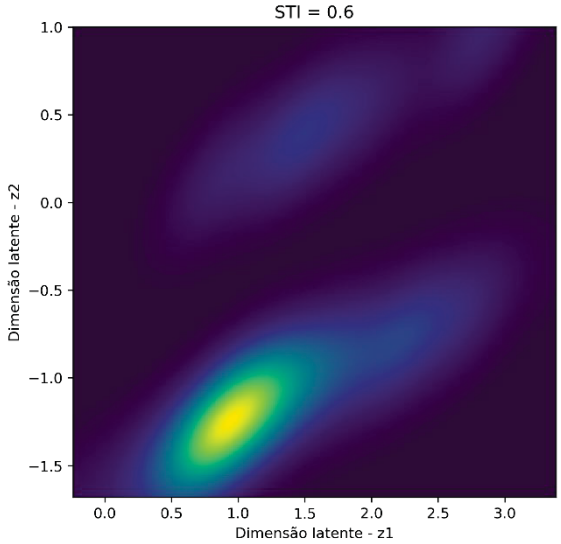
c) KDE 1D (z1 e z2) – STI [0.2,0.3[ - Domínio



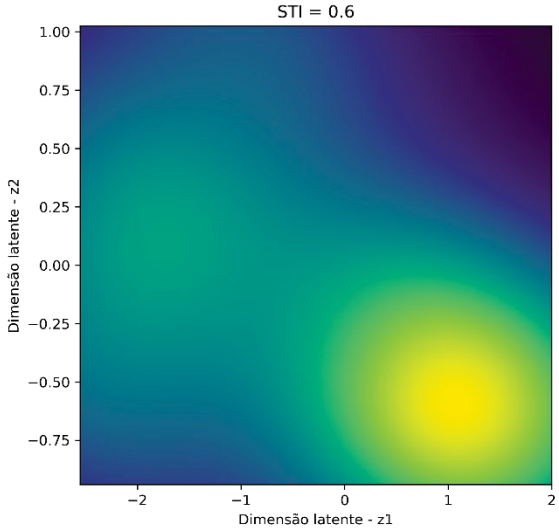

d) KDE 1D - (z1 e z2) – STI [0.2,0.3[ – Alvo




e) KDE 1D (z1 e z2) – STI [0.5,0.6[ - Domínio



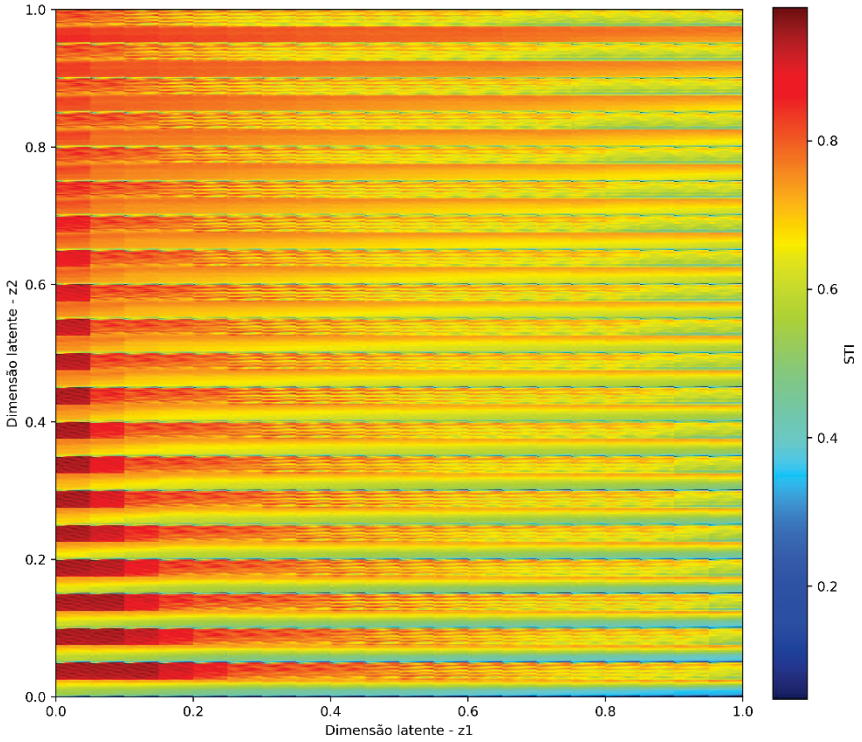

f) KDE 1D - (z1 e z2) – STI [0.5,0.6[ – Alvo




FONTE: O autor (2023).

De acordo com He, Mao e Chen (2015), no contexto de transferência de aprendizagem, um gráfico do KDE pode ser usado para comparar as distribuições dos domínios de origem e alvo antes e depois do cálculo da minimização da métrica MMD. Dessa forma, se as distribuições estiverem alinhadas, os gráficos do KDE dos domínios de origem e alvo apresentam semelhança. Por outro lado, se as distribuições não estiverem alinhadas, os gráficos do KDE serão discordantes, indicando uma discrepância entre as duas distribuições. Além disso, a FIGURA 33 destaca o conhecimento de auto-organização topológica obtido das coordenadas do espaço latente.

FIGURA 33 – ESPAÇO LATENTE: RELAÇÃO ENTRE RUÍDO DE FUNDO E TEMPO DE REVERBERAÇÃO SOBRE O STI




FONTE: O autor (2023).

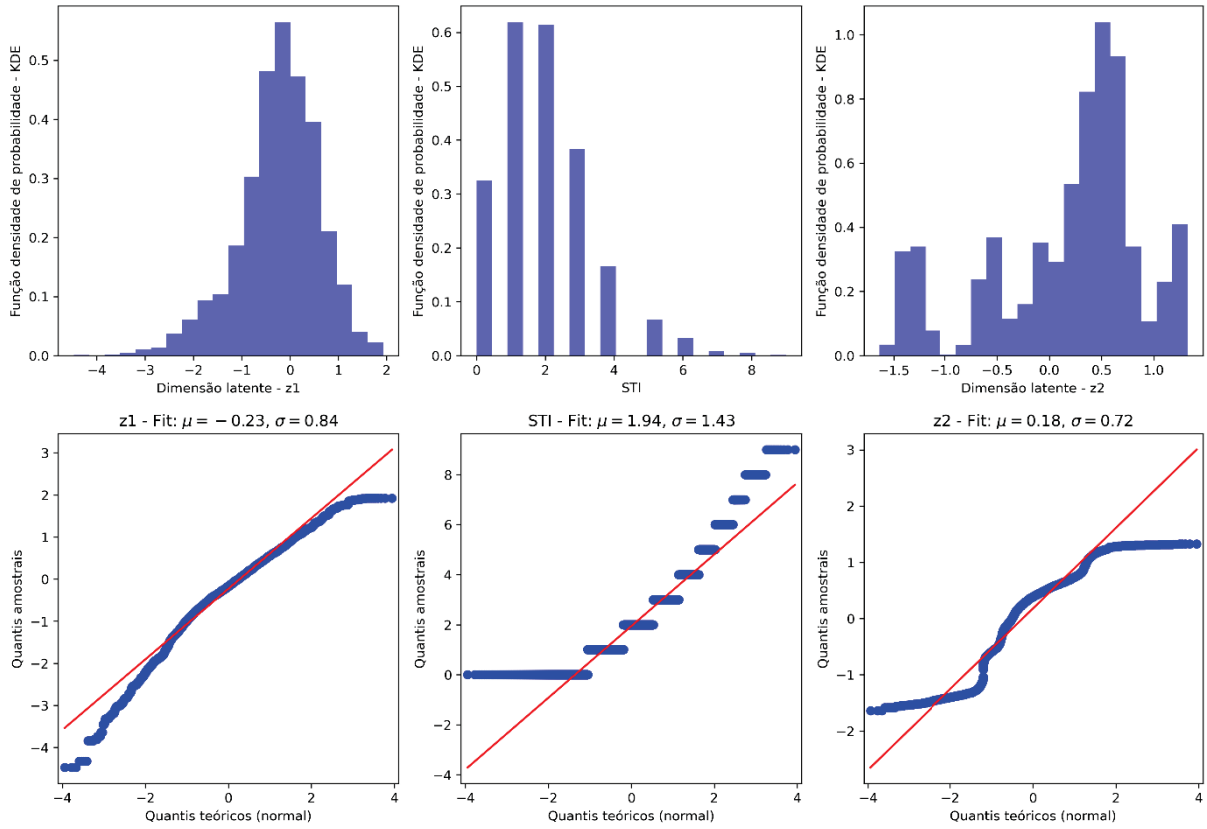
Conforme a FIGURA 33, observa-se que as variações do espaço de entrada composto pelas amostras do ruído de fundo e respostas impulsivas foram organizadas no espaço latente de acordo com as classes STI. Em outras palavras, à medida que o STI aumentou, os valores mais altos de BGN (ruído de fundo) foram atribuídos ao lado esquerdo do gráfico, enquanto os valores mais baixos foram atribuídos ao lado direito do gráfico em função do RIR (resposta impulsiva). Essa é a mesma dinâmica de variação do STI em função do ruído de fundo e tempo de reverberação.

Esses resultados são consistentes com a literatura corrente e demonstram que o espaço latente reflete as distribuições de classes mostradas na FIGURA 29. Além disso, fica evidente a distribuição desbalanceada de classes entre os conjuntos de dados, com mais classes sendo atribuídas ao intervalo entre zero e um. Este resultado concorda com as observações de Bistafa e Bradley (2000).

## 4.4 IMPLEMENTAÇÃO DA TRANSFERÊNCIA DE APRENDIZAGEM

Previamente na FIGURA 29 foram consolidadas as distribuições no espaço latente gerado pela rede VAE, nessa foram estratificadas as distribuições em função do STI nas faixas de (k = 0,1; 0,3; 0,6 e 0,9 do STI), restritas a duas dimensões, a saber, z1 e z2. Especificamente, na FIGURA 34, tem-se a representação do histograma da distribuição do codificador variacional, ao longo das dimensões latentes, z1 e z2. Esse resultado é independente da estratificação dos valores do STI.

FIGURA 34 – ESTIMATIVA DE DENSIDADE KERNEL DO ESPAÇO LATENTE Z1, STI E Z2




FONTE: O autor (2023).

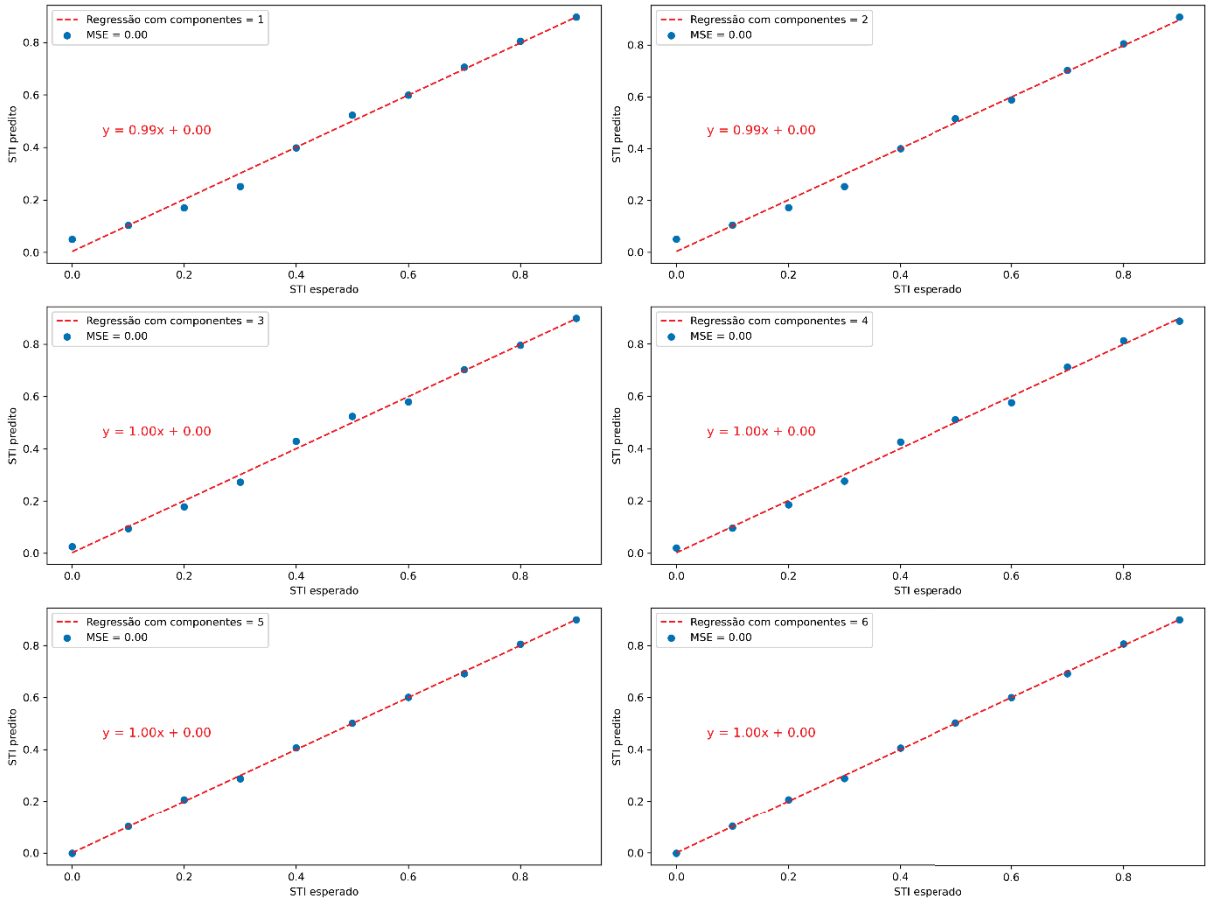
LEGENDA: Gráficos em primeira linha representam a estimativa de densidade kernel (KDE) para as coordenadas latentes (z1 e z2), enquanto os gráficos da segunda linha representam o Gráfico Quantil-Quantil, com um ajuste de distribuição normal.

Ao inspecionar a FIGURA 34(b), tem-se o histograma do STI para o conjunto de origem. As FIGURAS 34(a)-(c) representam a localização no espaço das probabilidades para as coordenadas z1 e z2. Nota-se que se identifica o centro

distribuições z1 e z2 como -0,23  e 0,18, respectivamente. Logo, como corolário as coordenadas z1 e z2, dada a diferença dos pontos centrais, são capazes de indicar um valor do STI somente com base na posição do espaço latente. A visualização bidimensional do espaço latente foi mostrada na FIGURA 29.

A FIGURA 35 mostra o ajuste do modelo KPCR aplicado no espaço latente, evidenciando a calibração deste modelo.

FIGURA 35 – CURVA DE CALIBRAÇÃO DO MODELO KPCR APLICADO NO ESPAÇO LATENTE



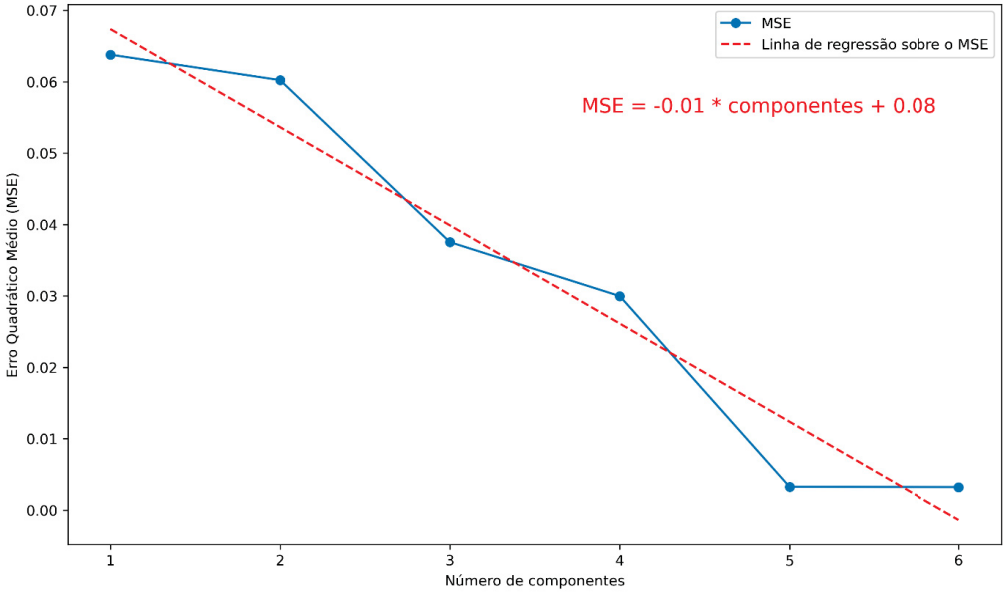

FONTE: O autor  (2023).

LEGENDA: Gráfico do ajuste do modelo de regressão via KPCR e comparação das curvas de calibração em função da quantidade de componentes considerados na estimativa do STI.

Matematicamente a obtenção do valor do STI em função da posição (z1, z2), no espaço latente, foi obtida via a regressão de componentes principais via núcleo, KPCR, que posteriormente foi aplicada no algoritmo MVPAnP.

Adicionalmente, a FIGURA 35 mostra o efeito do número de componentes principais sobre a qualidade do ajuste do KPCR. A qualidade do ajuste foi aferida pela medida do Erro Quadrático Médio (MSE) como a métrica do mapeamento, de acordo com a Eq. 67. Dado o espaço de probabilidade de distribuição sobre as coordenadas latentes, construiu-se uma linha de regressão linear usando KPCR para o conjunto de dados de treinamento.

FIGURA 36 – AVALIAÇÃO DO EQM *VERSUS* NÚMERO DE COMPONENTES PRINCIPAIS




FONTE: O autor (2023).

LEGENDA: Gráfico do Erro Quadrático Médio em função do número de componentes principais.

Para avaliar a qualidade do ajuste com o algoritmo MVPAnP, em relação aos métodos de referência, foram comparadas configurações de regressão via o modelo da rede neural convolucional profunda no domínio de alvo, com e sem TF, aplicando a metodologia de validação cruzada (CV = 10) e com os hiperparâmetros da rede 1D CONV NET apresentados na TABELA 3. Os resultados mostraram uma melhora no problema de regressão usando a abordagem proposta neste trabalho, conforme a TABELA 10.

TABELA 10 – VALIDAÇÃO CRUZADA DO MODELO 1D CONV NET SEM TF E COM TF (MVPAnP)

| Rodada | Aplicou TF? | Função de ativação | Dropout | Filtros | kernel | strides | Duração (s) | Função de perda |
|---|---|---|---|---|---|---|---|---|
| 1 | Sem | relu | 0,23 | 64 | 3 | 2 | 00:38,40 | 0,070 |
| 2 | Sem | relu | 0,27 | 32 | 3 | 1 | 00:36,58 | 0,070 |
| 3 | Sem | relu | 0,23 | 32 | 5 | 2 | 00:36,56 | 0,285 |
| 4 | Sem | swish | 0,29 | 64 | 3 | 1 | 00:37,14 | 0,114 |
| 5 | Sem | relu | 0,29 | 32 | 3 | 2 | 00:36,46 | 0,251 |
| 6 | Sem | relu | 0,29 | 32 | 5 | 2 | 00:36,49 | 0,285 |
| 7 | Sem | relu | 0,18 | 32 | 5 | 2 | 00:36,32 | 0,069 |
| 8 | Sem | sigmoid | 0,24 | 64 | 5 | 2 | 00:36,64 | 0,083 |
| 9 | Sem | swish | 0,25 | 32 | 5 | 2 | 00:37,29 | 0,285 |
| 10 | Sem | swish | 0,26 | 32 | 3 | 2 | 00:37,04 | 0,069 |
| 1 | Com | sigmoid | 0,16 | 64 | 3 | 2 | 01:41,15 | -0,041 |
| 2 | Com | relu | 0,11 | 64 | 5 | 1 | 01:40,28 | -0,036 |
| 3 | Com | swish | 0,10 | 64 | 3 | 2 | 01:44,12 | -0,052 |
| 4 | Com | sigmoid | 0,22 | 64 | 3 | 1 | 01:42,60 | -0,041 |
| 5 | Com | swish | 0,11 | 64 | 5 | 2 | 01:42,33 | -0,041 |
| 6 | Com | sigmoid | 0,27 | 32 | 5 | 1 | 01:42,50 | -0,041 |
| 7 | Com | relu | 0,13 | 64 | 5 | 2 | 01:42,61 | 0,161 |
| 8 | Com | swish | 0,24 | 32 | 3 | 2 | 01:43,48 | -0,050 |
| 9 | Com | sigmoid | 0,24 | 32 | 5 | 1 | 01:39,09 | -0,041 |
| 10 | Com | sigmoid | 0,17 | 64 | 5 | 1 | 01:40,70 | -0,041 |

NOTA 1: comparação entre diferentes combinações dos hiperparâmetros, Função de ativação (sigmoid, swish e ReLu); Dropout (0,1; 0,2; ou 0,3), Dimensionalidade dos filtros convolucionais (34 ou 64), Dimensão do kernel ( 3 ou 5) , Strides do Kernel (1 ou 2).

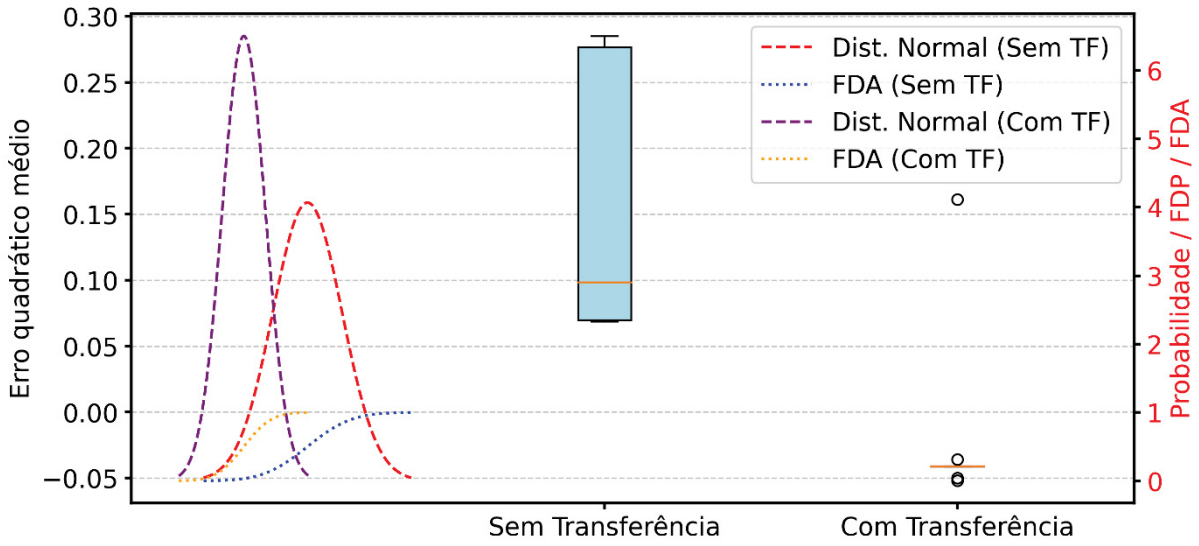
NOTA 2: * - A função de perda sem transferência de aprendizagem foi o MSE e com transferência foi a função MVPAnP_Loss conforme a Eq. 64.

FONTE: O autor (2023).

A melhoria com o uso do método KPCR foi 6 vezes maior, em termos da média do erro MSE para a validação cruzada, quando realiza-se o cálculo da razão da média dos MSE, nos cenários com e sem TF, com os valores de médios absolutos de 0,147 e 0,022, ambos em termos absolutos. Essas constatações demonstram a eficácia dessa abordagem proposta para melhorar a precisão da regressão.

Os resultados tabulares da TABELA 10 são destacados estatisticamente para a FIGURA 37, onde compara-se graficamente o desempenho do método proposto com e sem transferência de aprendizagem via o gráfico *box-plot*, e a distribuições acumuladas do erro quadrático conforme as rodadas da validação cruzada.

FIGURA 37 – COMPARAÇÃO DO EQM/MVPanP DO MODELO 1D CONV NET COM E SEM TF




FONTE: O autor (2023).

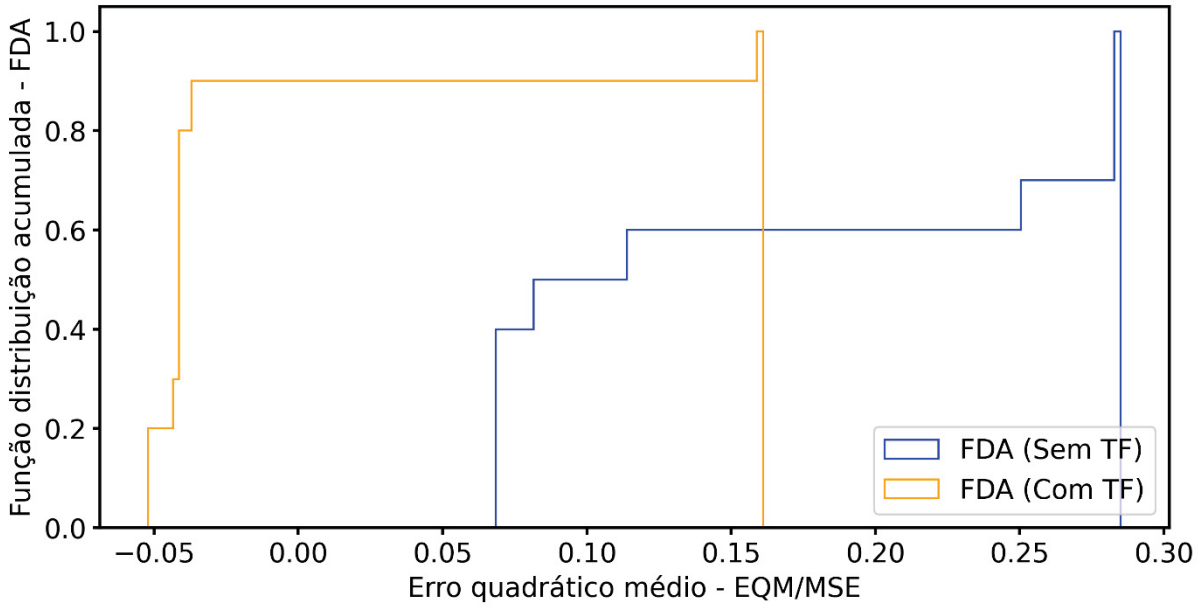
LEGENDA: Função densidade de probabilidade (FDP) e Função Densidade Acumulada.

NOTA: Os valores negativos do EQM no cenário com TF são devidos a minoração na função de perda MVPAnP com a divergência KL, conforme mostra a Eq. 64. Nesse caso, a expressão do EQM fica em função da injeção do MVPAnP, ou seja, EQM(MVPAnP).

Mediante o exposto, na seção 3.6, propôs-se, neste trabalho, um novo algoritmo de transferência aprendizagem profunda baseado em observações não rotuladas, esse algoritmo foi denominado de MVPAnP. O método proposto nesta tese é baseado na ideia de impor a distribuição a posteriori agregada, para estar próxima da normal padrão a priori. Isso contrasta com a abordagem usada no VAE tradicional conforme que, obriga cada amostra posterior estar próxima da a priori (LI et al., 2020; BARRETO et al., 2022).

Dado esta contextualização, a validação do algoritmo MVPAnP foi realizada, com base no teste ANOVA, como hipótese nula, com ($p < 0,01$) e 95% de certeza, que modelo MVPAnP usando KPCR melhorou significativamente a precisão do modelo no conjunto de dados de teste e aplicando o modelo da rede 1D CONV NET como referência para a implementação da função custo acoplada com o algoritmo MVPAnP. A FIGURA 38 consolida o gráfico da distribuição acumulada para os cenários com e sem transferência de aprendizagem.

FIGURA 38 – DISTRIBUIÇÃO CUMULATIVA DO EQM COM E SEM TF




FONTE: O autor (2023).

. Portanto, a FIGURA 38 demonstra a comparação entre as distribuições cumulativas para o treinamento da rede 1D CONV NET, apresentados na TABELA 3, com e sem transferência de aprendizagem. Nesta mostra-se que o algoritmo MVPAnP apresentou melhor desempenho em conjuntos de dados de alvo e que gerou a possibilidade uma interpretação do termo de regularização dos espaços latentes via a taxativa correspondência de valores de STI com a respectiva distribuição das coordenadas latentes (z1, z2), conforme previamente mostrado na FIGURA 29.

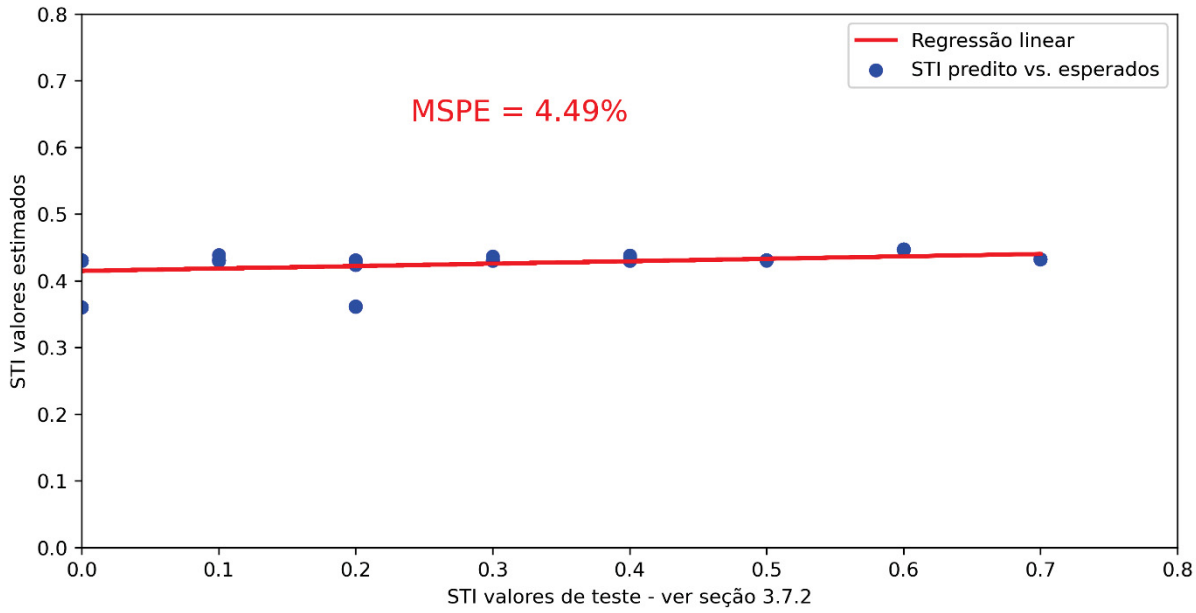
## 4.5 ACÚSTICA DE SALAS E PREDIÇÃO DO STI VIA ALGORITMO MVPANP

Nessa tese, foi proposto o uso de um modelo padrão VAE, que após treinada gerou uma distribuição posterior agregada no espaço latente de origem que esteve próximo uma distribuição dos dados do conjunto alvo por meio da regressão de componentes principais (KPCR). Dessa forma, o método MVPAnP atuou como uma *proxy* de correspondência de rótulos tomando como base a distância entre a distribuição posterior latente conhecida, que é mais fácil de minimizar diretamente e incluir na função custo de treinamento, que no caso forma os modelos de redes convolucionais profundas unidimensionais padrões (*vanilla*).

Dada essa contextualização, na literatura foi verificado algumas limitações entre as medições e a correspondente modelagem do STI via métodos tradicionais. Assim, diversas dessas complexidades e limitações inerentes à medição do STI foram destacadas, tais como as fontes clássicas de erros de medição conforme elencadas na IEC 60268-16 (IEC, 2011). Isso inclui fenômenos acústicos como diretividade da fonte, equalização do sinal, mascaramento e ressonância harmônica associada aos modos de vibrar da sala.

Para prover clareza sobre a interpretação do DMP de 3,00%, em relação aos erros de predição do STI, tem-se confeccionado na FIGURA 39, a curva de calibração, com os respectivos erros médios percentuais MSPE = 4,49% e RMPSE 2,23%. Esses resultados demonstram a eficácia do algoritmo proposto em melhorar o desempenho dos modelos de aprendizado de transferência profunda.

FIGURA 39 – CURVA DE CALIBRAÇÃO DOS VALORES PREDITOS STI VIA 1D CONV NET + TF




FONTE: O autor (2023).

Em termos de modelagem e erro aceitáveis para diversos modelos, a abordagem de Bradley, Reich e Norcross (1999) estabeleceu o conceito de Diferença Minimamente Perceptível (DMP) para avaliar a aceitabilidade dos erros de medição/modelagem, sendo este, no caso do STI estabelecido como 0,03.

Com o algoritmo MVPAnP acoplado numa rede neural profunda convolucional foi determinado um Erro Quadrático Médio (EQM) de 0,09. Em termos de escala, para n = 600 salas. Salienta-se que um erro de 0,09 é a soma quadrática do resíduo ponderada pela quantidade de amostras, esse valor, portanto, médio para é muito menor que o DMP = 0,03.

Doravante, ao comparar a escala do erro da saída do modelo 1D CONV NET + MVPAnP delineada no parágrafo anterior, constatou-se uma atinência com erros aceitáveis. Isso posto, na literatura há uma vasta gama de modelos para predição do STI, que buscaram também contornar as limitações apresentadas na seção 2.5.1.

Assim, para sumarizar, as demais abordagens de modelagem incluíram técnicas tradicionais de regressão baseadas em variados descritores acústicos (TANG; YEUNG, 2004; ESCOBAR; MORILLAS, 2015; NOWOŚWIAT; OLECHOWSKA, 2016; LECCESE, ROCCA, SALVADORI, 2018), bem como modelos mais avançados baseados em redes neurais profundas (Seetharaman et al., 2018) e uma metodologia híbrida proposta por Unoki et al. (2017). Uma vantagem do algoritmo MVPAnP, é que ele pode ser utilizado nessas mesmas modelagens anteriores, uma vez que este é independente do domínio de análise e pode ser extrapolado.

Outrossim, o algoritmo proposto para a transferência de aprendizagem via MVPAnP, demonstra algumas melhorias substanciais em comparação com os métodos tradicionais, principalmente quando se abarca a possibilidade que essa modelagem pode ser empregada em ambientes com ruído de fundo não estacionário. Além disso, destaca sua aplicabilidade prática em avaliações diagnósticas do STI em salas de aula, oferecendo uma solução mais acessível em termos de instrumentação necessária. As conclusões ressaltam a viabilidade do método MVPAnP em outros campos de estudo que buscam a transferência de aprendizagem (LI et al., 2020; BARRETO et al., 2022).

# 5 CONSIDERAÇÕES FINAIS

Este estudo aplicou o modelo VAE para analisar conjuntos de dados e avaliar sua interpretabilidade. O número de dimensões foi identificado como um parâmetro crítico para o modelo VAE, pois afeta a compressão aplicada aos dados. Além disso, técnicas de IA explicáveis foram usadas para avaliar a interpretabilidade do espaço latente calculado pelo modelo. Mostrou-se que a transferência de aprendizagem utilizando o algoritmo MVPAnP atingiu seus objetivos.

## 5.1 CONCLUSÕES

O modelo VAE foi treinado e otimizado com hiperparâmetros, e a transferência de aprendizagem foi aplicada para determinar o estado representacional de cada conjunto de dados de origem/alvo. O desempenho do modelo foi avaliado com cálculos de função de perda. A projeção PCA dos dados mostrou que o PCA não capturou de forma suficiente a variância dos dados, enquanto a projeção VAE das classes de dados permitiu uma compreensão mais completa do comportamento do modelo.

A capacidade do modelo de representar e generalizar os valores do STI no espaço latente foi avaliada com dois conjuntos, um para treinamento e testes e outro usado como validação (Alvo). Os resultados mostraram um nível proeminente de concordância entre os conjuntos de treinamento e teste, sugerindo que o modelo é capaz de generalizar, face aos dados apresentados durante o treinamento. Portanto, o método de TF composto de redes neurais profundas possuirá a capacidade de adaptar as mais diversas salas de aula.

Conclui-se que o modelo de transferência de aprendizagem foi capaz de predizer o STI para ruído de fundo não estacionário. Nesse âmbito foi desenvolvido um novo método de transferência de aprendizagem profunda. Esse método, denominado de transferência de aprendizagem via Minimização Variacional Projetiva-Adaptativa não Paramétrica (MVPAnP), possui um caráter multifacetário. Ele, portanto, poderá ser aplicado em outros campos de estudo que objetivam realizar a transferência de aprendizagem.

Finalmente, em vista dos custos e da consequente inviabilidade orçamentária para aquisição da instrumentação requerida para a medição do STI, o algoritmo MVPAnP poderá ser utilizado como uma ferramenta de suporte para uma avaliação diagnóstica do STI em salas de aula, usando apenas a instrumentação que é associada às  medições do TR.

## 5.2 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Tendo como base o desenvolvimento deste trabalho, são formuladas as seguintes sugestões de trabalho futuro:

- Aplicar, para a criação dos dados sintéticos, o software ODEON;
- Desenvolver e realizar a curadoria de um banco de dados de respostas impulsivas e ruído de fundo em diferentes tipos de salas, uma vez que neste trabalho foram somente utilizados dados de medições de RIR das salas de aula da Universidade Federal do Paraná;
- Desenvolver o uso de algoritmos generativos para geração dos modelos.

# REFERÊNCIAS

ABDELJABER, O. et al. Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. **Journal of Sound and Vibration**, v. 388, p. 154-170, 2017.

ALLEN, J. B.; BERKLEY, D. A. Image method for efficiently simulating small-room acoustics. **The Journal of the Acoustical Society of America**, v. 65, n. 4, p. 943-950, 1979.

AKIBA, T. et al. "Optuna: A next-generation hyperparameter optimization framework." **In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining**, p. 2623-2631. 2019.

AMERICAN NATIONAL STANDARD INSTITUTE (ANSI). **ANSI/ASA S12.60**-2010/PART 1: Acoustical Performance criteria, Design Requirements, and guidelines for Schools, Part 1: Permanent Schools. Acoustic Society of America, 2010.

ANSAY, S.; ZANNIN, P. H. T. Using the Parameters of Definition, D50, and Reverberation Time, RT, to Investigate the Acoustic Quality of Classrooms. **Canadian Acoustics**, v. 44, n. 4, p. 6-11, 2016.

ARNOLD, T B.; EMERSON, J. W. Nonparametric goodness-of-fit tests for discrete null distributions. **R Journal**, v. 3, n. 2, 2011.

BARRETO, F. et al. S. Unsupervised Domain Adaptation using Maximum Mean Covariance Discrepancy and Variational Autoencoder. **International Journal of Advanced Computer Science and** Applications, v.13, n. 6, 2022.

BARTLETT, M.S, Periodogram Analysis and Continuous Spectra, **Biometrika**, v. 37, p. 1-16, 1950.

BASNER, M. et al. Auditory and non-auditory effects of noise on health. **The Lancet**, v. 383, n. 9925, p. 1325-1332, 2014.

BISTAFA, S. R.; BRADLEY, J. S. Reverberation time and maximum background-noise level for classrooms from a comparative study of speech intelligibility metrics. **The Journal of the Acoustical Society of America**, v. 107, n. 2, p. 861-875, 2000.

BOUWMANS, T. et al. Deep neural network concepts for background subtraction: A systematic review and comparative evaluation. **Neural Networks**, v. 117, p. 8-66, 2019.

BRADLEY, J. S.; REICH, R.; NORCROSS, S. G. A just noticeable difference in C50 for speech. **Applied Acoustics**, v. 58, n. 2, p. 99-108, 1999.

CANCHUMUNI, S. W. A; EMERICK, A. A.; PACHECO, M. A. C. Towards a robust parameterization for conditioning facies models using deep variational autoencoders and ensemble smoother. **Computers & Geosciences**, v. 128, p. 87-102, 2019.

CHOI, Y.-Ji. Evaluation of acoustical conditions for speech communication in active university classrooms. **Applied Acoustics**, v. 159, p. 107089, 2020.

CHOLLET, F. al. **Keras: The python deep learning library**. ascl, p. ascl: 1806.022, 2018.

CHOLLET, F. **Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek**. MITP-Verlags GmbH & Co. KG, 2018.

CONNOLLY, D. et al. The effects of classroom noise on the reading comprehension of adolescents. **The Journal of the Acoustical Society of America**, v. 145, n. 1, p. 372-381, 2019.

CRUCIANI, F. et al. Feature learning for Human Activity Recognition using Convolutional Neural Networks. **CCF Transactions on Pervasive Computing and Interaction,** v. 2, n. 1, p. 18-32, 2020.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. Mathematics of control, **Signals and Systems**, v. 2, n. 4, p. 303-314, 1989.

DEPARTMENT FOR EDUCATION (DFE), **BB93**: Building Bulletin 93 - Acoustic design of schools: performance standards. London. 2015.

DEUTSCHES INSTITUT FÜR NORMUNG (DIN) - DIN 18041.2004. **Hörsamkeit in kleinen bis mittelgroßen Räumen**. [Acoustical quality in small to medium-sized rooms] Germany, Berlin, Beuth Verlag, 2004.

DHILLON, A.; VERMA, G. K. Convolutional neural network: a review of models, methodologies and applications to object detection. **Progress in Artificial Intelligence**, v. 9, n. 2, p. 85-112, 2020.

DOCKRELL, J. E.; SHIELD, B. M. Acoustical barriers in classrooms: The impact of noise on performance in the classroom. **British Educational Research Journal**, v. 32, n. 3, p. 509-525, 2006.

ESCOBAR, V. G; MORILLAS, J. M B. Analysis of intelligibility and reverberation time recommendations in educational rooms. **Applied Acoustics**, v. 96, p. 1-10, 2015.

EVANS, L. C. **Partial differential equations and Monge-Kantorovich mass transfer**. Current Developments in Mathematics, v. 1997, n. 1, p. 65-126, 1997.

FINNISH STANDARDS ASSOCIATION (SFS), **SFS 5907**: en ACOUSTIC CLASSIFICATION OF SPACES IN BUILDINGS. SUOMEN STANDARDISOIMISLIITTO. 2006.

GUO, L. et al. Effects of environmental noise exposure on DNA methylation in the brain and metabolic health. **Environmental research**, v. 153, p. 73-82, 2017.

GUPTA, N. et al. Evolutionary Artificial Neural Networks: Comparative Study on State-of-the-Art Optimizers. **In: Frontier Applications of Nature Inspired Computation. Springer**, Singapore, 2020. p. 302-318.

HARRIS, C. M. **Handbook of Acoustical Measurements and Noise Control**. New York: McGraw-Hill, 1991.

HE, Y., MAO, W; CHEN; Y. CHEN, Nonlinear Metric Learning with Kernel Density Estimation. **IEEE Transactions on Knowledge and Data Engineering**, v. 27, n. 6, p. 1602-1614, 2015.

HERRMANN, J. O. 117 p. **Medições e simulações de índice de transmissão de fala (STI) definição (D50) e tempo de reverberação (Tr) em salas de aula.** Dissertação (mestrado) - Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia Ambiental, Curitiba -PR, 2018.

HOUTGAST, T., STEENEKEN, H.J.M. The modulation transfer function in room acoustics as a predictor of speech intelligibility. **Acta Acustica united with Acustica**,28, pp. 66-73., 1973

HOUTGAST, T.; STEENEKEN, H. JM; PLOMP, R. Predicting speech intelligibility in rooms from the modulation transfer function. I. General room acoustics. **Acta Acustica united with Acustica**, v. 46, n. 1, p. 60-72, 1980.

HULVA, A. et al. Mapping speech transmission index (STI) and background noise in university classrooms. **The Journal of the Acoustical Society of America**, v. 141, n. 5, p. 3481-3481, 2017.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, Joaquin. Automated machine learning: methods, systems, challenges. **Springer Nature**, 2019.

INTERNATIONAL ELECTROTECHNICAL COMMISSION (IEC). **IEC 60268-16:** Sound system equipment – Part 16: Objective rating of speech intelligibility by speech transmission index. Switzerland, 2011.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). **ISO 3382-2:** Acoustics -- Measurement of room acoustic parameters -- Part 2: Reverberation time in ordinary rooms, Switzerland, 2008.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). **ISO 9921:** Ergonomics – Assessment of speech communication. Switzerland, 2003.

JOHN S. T., NELLO C. **Kernel Methods for Pattern Analysis**. Cambridge University Press, New York, NY, USA, 2004.

JOLLIFFE, I. T. A note on the use of principal components in regression. **Journal of the Royal Statistical Society: Series C (Applied Statistics),** v. 31, n. 3, p. 300-303, 1982.

KHALIL, M. et al. An End-to-End Multi-Level Wavelet Convolutional Neural Networks for Heart Diseases Diagnosis. **Neurocomputing**, v. 417, p. 187-201, 2020.

KINGMA, D. P.; BA, Jimmy. Adam: **A method for stochastic optimization**. arXiv preprint arXiv:1412.6980, 2014.

KINGMA, D. P.; WELLING, M. **Auto-encoding variational bayes**. arXiv preprint arXiv:1312.6114, 2013.

KIRANYAZ, S. et al. 1-d convolutional neural networks for signal processing applications**. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).** IEEE, p. 8360-8364, 2019.

KIRANYAZ, S.; INCE, T.; GABBOUJ, M. Real-time patient-specific ECG classification by 1-D convolutional neural networks. **IEEE Transactions on Biomedical Engineering,** v. 63, n. 3, p. 664-675, 2015.

KLATTE, M.; BERGSTRÖM, K.; LACHMANN, T. Does noise affect learning? A short review on noise effects on cognitive performance in children. **Frontiers in Psychology**, v. 4, p. 578, 2013.

KO, T. et al. A study on data augmentation of reverberant speech for robust speech recognition. **In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).** IEEE, p. 5220-5224, 2017.

KOLMOGOROV, A. **Sulla determinazione empirica di una legge didistribuzione." Giorn Dell'inst Ital Degli, Att,** 4, 89-91, 1933.

KOLMOGOROV, A. On the Empirical Determination of a Distribution Function**. In: Breakthroughs in Statistics**. Springer Series in Statistics, Springer, New York, NY., 106-113, 1992.

KUHN, M.; JOHNSON, K. **Feature engineering and selection: A practical approach for predictive models**. CRC Press, 2019.

LANE, H.; TRANEL, B. The Lombard sign and the role of hearing in speech. **Journal of Speech and Hearing Research,** v. 14, n. 4, p. 677-709, 1971.

LE, L.; PATTERSON, A.; WHITE, M. Supervised autoencoders: Improving generalization performance with unsupervised regularizers. **In: Advances in Neural Information Processing Systems**. p. 107-117, 2018.

LECCESE, F.; ROCCA, M.; SALVADORI, G. Fast Estimation of Speech Transmission Index using the Reverberation Time: Comparison between predictive

equations for educational rooms of different sizes. **Applied Acoustics**, v. 140, p. 143-149, 2018.

LEHMANN, E.; JOHANSSON, A. M.; NORDHOLM, Sven. Reverberation-time prediction method for room impulse responses simulated with the image-source model. **In: 2007 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics**. p. 159-162. IEEE, 2007.

LEVANDOSKI, G; ZANNIN, P. H. T. Quality of Life and Acoustic Comfort in Educational Environments of Curitiba, Brazil. **Journal of Voice**, 2020.

LI, F. et al. Feature extraction and classification of heart sound using 1D convolutional neural networks. **EURASIP Journal on Advances in Signal Processing**, v. 2019, n. 1, p. 59, 2019.

LI, X.; GRANDVALET, Y.; DAVOINE, F. A baseline regularization scheme for transfer learning with convolutional neural networks. **Pattern Recognition**, v. 98, p. 107049, 2020.

LI, Y. et al. Collapse susceptibility assessment using a support vector machine compared with back-propagation and radial basis function neural networks. **Geomatics, Natural Hazards and Risk**, v. 11, n. 1, p. 510-534, 2020.

LIU, H. et al. The speech intelligibility and applicability of the speech transmission index in large spaces. **Applied Acoustics**, v. 167, p. 107400, 2020.

LIU, L. et al. End-to-end binary representation learning via direct binary embedding. **In: 2017 IEEE International Conference on Image Processing (ICIP).** IEEE, p. 1257-1261, 2017.

LIU, X. et al. Fault diagnosis of rotating machinery under noisy environment conditions based on a 1-D convolutional autoencoder and 1-D convolutional neural network. **Sensors**, v. 19, n. 4, p. 972, 2019.

LONG, M. et al. Transfer feature learning with joint distribution adaptation. **In: Proceedings of the IEEE International Conference on Computer Vision**. p. 2200-2207, 2013.

LONGONI, H. C. et al. Speech transmission index variation due to ventilating and air-conditioning system in university classrooms. **In: Proceedings of Meetings on Acoustics 22ICA**. Acoustical Society of America, ASA, p. 015024, 2016.

LU, Y. et al. Bayesian optimized deep convolutional network for bearing diagnosis. **International Journal of Advanced Manufacturing Technology**, 2020.

MARSHALL, L. G. An acoustics measurement program for evaluating auditoriums based on the early/late sound energy ratio. **The Journal of the Acoustical Society of America,** v. 96, n. 4, p. 2251-2261, 1994.

MASSONNIÉ, J. et al. Is Classroom Noise Always Bad for Children? The Contribution of Age and Selective Attention to Creative Performance in Noise. **Frontiers in Psychology**, v. 10, p. 381, 2019.

MCGARRIGLE, R. et al. Behavioral measures of listening effort in school-age children: Examining the effects of signal-to-noise ratio, hearing loss, and amplification. **Ear and Hearing**, v. 40, n. 2, p. 381-392, 2019.

MEALINGS, K. T. et al. Investigating the acoustics of a sample of open plan and enclosed Kindergarten classrooms in Australia. **Applied Acoustics**, v. 100, p. 95-105, 2015.

MIKULSKI, W.; RADOSZ, J. Acoustics of classrooms in primary schools-results of the reverberation time and the speech transmission index assessments in selected buildings. **Archives of Acoustics**, v. 36, n. 4, p. 777-793, 2011.

MINICHILLI, F. et al. Annoyance judgment and measurements of environmental noise: A focus on Italian secondary schools. **International Journal of Environmental Research and Public Health**, v. 15, n. 2, p. 208, 2018.

MON, A. N.; PA PA, W.; THU, Y. K. Improving Myanmar Automatic Speech Recognition with Optimization of Convolutional Neural Network Parameters. **International Journal on Natural Language Computing (IJNLC)** Vol, v. 7, 2018.

NASCIMENTO, E. O. d. **Influência dos parâmetros acústicos *in situ* na avaliação da inteligibilidade da fala em salas de aula via redes neurais artificiais**. Setor de Tecnologia. Programa de Pós-Graduação em Engenharia Mecânica, p. 161, 2019.

NOWOŚWIAT, A.; OLECHOWSKA, M. Fast estimation of speech transmission index using the reverberation time. **Applied Acoustics**, v. 102, p. 55-61, 2016.

PALAZ, D.; MAGIMAI-DOSS, M.; COLLOBERT, R. End-to-end acoustic modeling using convolutional neural networks for HMM-based automatic speech recognition. **Speech Communication**, v. 108, p. 15-32, 2019.

PAN, H. et al. An improved bearing fault diagnosis method using one dimensional CNN and LSTM. **Journal of Mechanical Engineering**, v. 64, n. 7-8, p. 443-452, 2018.

PAN, S. J.; YANG, Q. A survey on transfer learning. **IEEE Transactions on Knowledge and Data Engineering,** v. 22, n. 10, p. 1345-1359, 2009.

PELEGRÍN-GARCÍA D.; BRUNSKOG J.; RASMUSSEN B. Speaker-oriented classroom acoustics design guidelines in the context of current regulations in European countries. **Acta Acustica united with Acustica**, v. 100, p. 1073-1089, 2014.

PENGE, D. et al. A Novel Deeper One-Dimensional CNN With Residual Learning for Fault Diagnosis of Wheelset Bearings in High-Speed Trains. **IEEE Access**, v. 7, p. 10278 – 10293, 2018.

PHADKE, K. V. et al. Influence of noise resulting from the location and conditions of classrooms and schools in Upper Egypt on teachers' voices. **Journal of Voice**, v. 33, n. 5, p. 802. e1-802. e9, 2019.

POELITZ, C. **Projection based transfer learning.** In: Workshops at ECML. 2014.

PRODI, N.; VISENTIN, C. An experimental study of a time-frame implementation of the Speech Transmission Index in fluctuating speech-like noise conditions. **Applied Acoustics**, v. 152, p. 63-72, 2019.

PUGLISI, G. E. et al. Influence of classroom acoustics on the reading speed: A case study on Italian second-graders. **The Journal of the Acoustical Society of America**, v. 144, n. 2, p. EL144-EL149, 2018.

QI, W. et al. A fast and robust deep convolutional neural networks for complex human activity recognition using smartphone. **Sensors**, v. 19, n. 17, p. 3731, 2019.

RABELO, A. T. V. et al. Effect of classroom acoustics on the speech intelligibility of students. **In: CoDAS. Sociedade Brasileira de Fonoaudiologia**, p. 360-366, 2014.

RABIN, N. et al. Modeling and Analysis of Students' Performance Trajectories using Diffusion Maps and Kernel Two-Sample Tests. **Engineering Applications of Artificial Intelligence**, v. 85, p. 492-503, 2019.

RANTALA, L. M.; SALA, E. Effects of Classroom Acoustics on Teachers' Voices. **Building Acoustics**, v. 22, n. 3-4, p. 243-258, 2015.

REZENDE, D. J.; MOHAMED, S.; WIERSTRA, D. Stochastic backpropagation and approximate inference in deep generative models. **arXiv preprint arXiv**:1401.4082, 2014.

ROSIPAL, R. et al. Kernel PCA for feature extraction and de-noising in nonlinear regression. **Neural Computing & Applications**, v. 10, n. 3, p. 231-243, 2001.

ROSIPAL, R.; TREJO, L. J. Kernel partial least squares regression in reproducing Kernel Hilbert Space. **Journal of Machine Learning Research**, v. 2, n. 12, p. 97-123, 2001.

ROSIPAL, R.; TREJO, L. J.; CICHOCKI, A. **Kernel principal component regression with EM approach to nonlinear principal components extraction**. University of Paisley, 2000.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533-536, 1986.

SABINE, W. C. Reverberation. **The American Architect**, v. 4, 1900.

SAARELMA, J. et al. Audibility of dispersion error in room acoustic finite-difference time-domain simulation as a function of simulation distance. **The Journal of the Acoustical Society of America,** v. 139, n. 4, p. 1822-1832, 2016.

SCHELTER, S.; RUKAT, T.; BIESSMANN, Felix. Learning to Validate the Predictions of Black Box Classifiers on Unseen Data. **In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data**. p. 1289-1299, 2020.

SCHROEDER, M. R. Modulation transfer functions: Definition and measurement. **Acta Acustica united with Acustica**, v. 49, n. 3, p. 179-182, 1981.

SEETHARAMAN, P. et al. Blind estimation of the speech transmission index for speech quality prediction. **In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. IEEE, p. 591-595, 2018.

SHIELD, B. et al. Noise in open plan classrooms in primary schools: A review. **Noise and Health**, v. 12, n. 49, p. 225, 2010.

SHAWE-TAYLOR, J.; CRISTIANINI. Kernel methods for pattern analysis. **Cambridge University Press**. p. 462, 2004.

STEENEKEN, H. J. M; HOUTGAST, T. Phoneme-group specific octave-band weights in predicting speech intelligibility. **Speech Communication**, v. 38, n. 3-4, p. 399-411, 2002.

TANG, S. K.; YEUNG, M. H. Speech transmission index or rapid speech transmission index for classrooms? A designer's point of view. **Journal of Sound and Vibration,** 2004.

TANG, Z.; MENG, H.-Yu; MANOCHA, D. Low-frequency compensated synthetic impulse responses for improved far-field speech recognition**. In: ICASSP 2020-2020 IEEE International Conference on Acoustics**, Speech and Signal Processing (ICASSP). IEEE, 2020. p. 6974-6978.

TENENBAUM, R. A. et al. Auralization generated by modeling HRIRs with artificial neural networks and its validation using articulation tests. **Applied Acoustics**, v. 130, p. 260-269, 2018.

TIELEMAN, T.; HINTON, G. **Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning**, Lecture 6.5-rmsprop, 2012.

UNOKI, M. et al. Method of Blindly Estimating Speech Transmission Index in Noisy Reverberant Environments. **Journal of Information Hiding and Multimedia Signal Processing**, v. 8, n. 6, p. 1430-1445, 2017.

VAN DE POLL, M. K. et al. Disruption of writing by background speech: The role of speech transmission index. **Applied Acoustics**, v. 81, p. 15-18, 2014.

VAN SCHOONHOVEN, J.; RHEBERGEN, K. S.; DRESCHLER, W. A. Towards measuring the Speech Transmission Index in fluctuating noise: Accuracy and limitations. **The Journal of the Acoustical Society of America**, v. 141, n. 2, p. 818-827, 2017.

VAN SCHOONHOVEN, J.; RHEBERGEN, K. S.; DRESCHLER, W. A. The extended speech transmission index: Predicting speech intelligibility in fluctuating noise and reverberant rooms. **The Journal of the Acoustical Society of America**, v. 145, n. 3, p. 1178-1194, 2019.

VORLÄNDER, M. **Auralization: Fundamentals of acoustics, modelling, simulation, algorithms, and acoustic virtual reality**. RWTH, edition Series Editor RWTH Aachen University, Aachen, Germany, p. 355, 2007.

WANG, J. et al. A deep learning method for bearing fault diagnosis based on time-frequency image. **IEEE Access**, v. 7, p. 42373-42383, 2019.

WANG, J. et al. Discriminative Feature Alignment: Improving Transferability of Unsupervised Domain Adaptation by Gaussian-guided Latent Alignment. **arXiv preprint arXiv:2006.12770**, 2020.

WEISS, K. KHOSHGOFTAAR, T. M.; WANG, D. D. A survey of transfer learning. **Journal of Big Data**, v. 3, n. 1, p. 9, 2016.

WELCH, P. The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. **IEEE Transactions on Audio and Electroacoustics**, v. 15, n. 2, p. 70-73, 1967.

WEN, X. et al. Impacts of traffic noise on roadside secondary schools in a prototype large Chinese city. **Applied Acoustics**, v. 151, p. 153-163, 2019.

WIBOWO, A. Hybrid kernel principal component regression and penalty strategy of multiple adaptive genetic algorithms for estimating optimum parameters in abrasive waterjet machining. **Applied Soft Computing**, v. 62, p. 1102-1112, 2018.

WIBOWO, A.; YAMAMOTO, Y. A note on kernel principal component regression. **Computational Mathematics and Modeling,** v. 23, n. 3, p. 350-367, 2012.

WU, J.; CHEN, SP; LIU, X. Efficient hyperparameter optimization through model-based reinforcement learning. **Neurocomputing**, v. 409, p. 381-393, 2020.

XIAO, Bi. et al. Heart sounds classification using a novel 1-D convolutional neural network with extremely low parameter consumption. **Neurocomputing**, v. 392, p. 153-159, 2020.

XU, Z. et al. Identifying crashing fault residence based on cross project model. **In: 2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE).** IEEE, p. 183-194, 2019.

YOUNG-JI, C. H. O. I. Comparison of two types of combined measures, STI and U50, for predicting speech intelligibility in classrooms. **Archives of Acoustics**, v. 42, n. 3, p. 527–532, 2017.

YUAN, X.; GE, Z.; SONG, Z. Locally weighted kernel principal component regression model for soft sensing of nonlinear time-variant processes. **Industrial & Engineering Chemistry Research,** v. 53, n. 35, p. 13736-13749, 2014.

ZENG, M. et al. Convolutional neural networks for human activity recognition using mobile sensors. **In: 6th International Conference on Mobile Computing, Applications and Services**. IEEE, p. 197-205, 2014.

ZHAI, S. et al. Doubly convolutional neural networks. **In: Advances in neural Information Processing Systems**. p. 1082-1090. 2016.
ZHANG, Y., et al. Base-2 softmax function: Suitability for training and efficient hardware implementation. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v. 69, n. 9 , p. 3605-3618, 2022.

ZHAO, W. et al. A Novel Deep Neural Network for Robust Detection of Seizures Using EEG Signals. **Computational and Mathematical Methods in Medicine**, v. 2020, 2020.

ZHENG, H. et al. Cross-domain fault diagnosis using knowledge transfer strategy: A review. **IEEE Access**, v. 7, p. 129260-129290, 2019.

ZHU, P. et al. Experimental comparison of speech transmission index measurement in natural sound rooms and auditoria. **Applied Acoustics**, v. 165, p. 107326, 2020.

ZHUANG, F. et al. A comprehensive survey on transfer learning. **Proceedings of the IEEE**, 2019.

## ANEXO – CÓDIGO FONTE

**Autor:** Eriberto Oliveira do Nascimento
**Data de criação:** 17/02/2019
**Última modificação:** 27/10/2023
**Descrição:** Rede Neural Variacional Autocodificadora (VAE) e Transferência de Aprendizagem Profunda com função de perda personalizada.
**Descrição Longa:** Código fonte da tese - DESENVOLVIMENTO DA TRANSFERÊNCIA DE APRENDIZAGEM VIA REDES NEURAIS ARTIFICIAIS PROFUNDAS NA MODELAGEM DO ÍNDICE DE TRANSMISSÃO DA FALA

```
"""### 1 - Importa módulos """
# -*- coding: utf-8 -*-
import sys
import os
import glob
import math
import argparse
import librosa
import random
import time
import urllib
import itertools
import warnings
import pandas as pd
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
matplotlib.use('Agg')

from matplotlib.gridspec import GridSpec
from numpy import linalg as LA
from numpy import savez_compressed
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.graphics.gofplots import qqplot

import scipy
from scipy import stats
from scipy import signal
import scipy.stats as stats
from scipy.fft import fft
from scipy.stats import kde
from scipy.io import wavfile
from scipy.stats import norm
from scipy.stats import ks_2samp
from scipy.stats import multivariate_normal
from scipy.stats import dirichlet
from scipy.stats import norm, f_oneway, ks_2samp
from scipy.stats import kde
from scipy.stats import f_oneway

from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE, Isomap
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.gaussian_process.kernels import ConstantKernel, RBF, DotProduct
from sklearn.decomposition import KernelPCA
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.manifold import TSNE, Isomap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import KernelPCA
from sklearn.pipeline import Pipeline
from sklearn.metrics import r2_score

import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Reshape, GlobalAveragePooling1D
from keras.layers import Conv2D, MaxPooling2D, Conv1D, MaxPooling1D
from keras.utils import np_utils
from keras.layers import Input, Dense, Lambda
from keras.models import Model
from keras import backend as K
from keras import objectives
from keras.backend import clear_session
from keras.utils.vis_utils import plot_model
from keras.layers.core import Dropout, Activation
from keras.models import Sequential
from keras.utils import np_utils
from keras.losses import binary_crossentropy
from sklearn.model_selection import train_test_split
from keras.models import Model

import tensorflow as tf
from tensorflow.keras.optimizers import RMSprop

from itertools import combinations
from joblib import dump, load

import optuna
from optuna.visualization import plot_contour
from optuna.visualization import plot_edf
from optuna.visualization import plot_intermediate_values
from optuna.visualization import plot_optimization_history
from optuna.visualization import plot_parallel_coordinate
from optuna.visualization import plot_param_importances
from optuna.visualization import plot_slice

from imblearn.over_sampling import RandomOverSampler
from collections import Counter

import torch
from torch.distributions.multivariate_normal import MultivariateNormal

from statsmodels.graphics.gofplots import qqplot
from scipy.stats import f_oneway
from statsmodels.stats.multicomp import pairwise_tukeyhsd
SEED = 42
np.random.seed(SEED)

!pip install acoustics
from acoustics import Signal
import acoustics
```

**1 - GERAÇÃO SINTÉTICA DA RESPOSTA IMPULSIVA DA SALA (RIR)**

```
# !pip install pyroomacoustics==0.4.1
# !pip install matplotlib==3.2.2

"""
Este exemplo cria uma sala com tempo de reverberação especificado invertendo a fórmula de Sabine.
Isto resulta num tempo de reverberação ligeiramente maior do que o desejado.
A simulação é feita via o Método das Imagens.
"""

# import pyroomacoustics as pra

def RIR_generator(signal_excitation):

    methods = ["ism", "hybrid"]

    if __name__ == "__main__":

        parser = argparse.ArgumentParser(
            description="Simulates and adds reverberation to a dry sound sample."
        )
        parser.add_argument(
            "--method",
            "-m",
            choices=methods,
            default=methods[1],
            help="Simulation method to use",
        )

        args = parser.parse_args()

        # O tempo de reverberação desejado e as dimensões da sala
        rt60_tgt = 2*random.random() + 0.1; # [seconds]
        rt60_tgt =round(rt60_tgt,2)
        room_dim = random.sample(range(5, 20), 3); # [meters]
        room_dim[2] = int(np.array(random.sample(range(2, 5), 1)))

        print(room_dim)

        # importa um arquivo wav mono como sinal fonte
        # a frequência de amostragem deve corresponder a da sala
        fs, audio = wavfile.read(signal_excitation)

        # Inverte-se  a fórmula de Sabine para obter os parâmetros do simulador ISM
        e_absorption, max_order = pra.inverse_sabine(rt60_tgt, room_dim)

        # Cria a sala virtual
        if args.method == "ism":
            room = pra.ShoeBox(
                room_dim, fs=fs, materials=pra.Material(e_absorption), max_order=max_order
            )

        elif args.method == "hybrid":
            room = pra.ShoeBox(
                room_dim,
                fs=fs,
                materials=pra.Material(e_absorption),
```

```
        max_order=3,
        ray_tracing=True,
        air_absorption=True,
    )

fig, ax = room.plot()

# insere  a fonte virtual  na sala
source_pos = np.array(room_dim)/2;
source_pos[2] = 1.5
source_pos = source_pos.tolist()

room.add_source(source_pos, signal=audio, delay=0.5)

# define a localização dos microfones
mic1 =   np.round( np.array(source_pos )+ np.array( [source_pos[0]*0.4, 0, 0]), 2)
mic4 =   np.round( np.array(source_pos )+ np.array( [0, -source_pos[1]/2.5, 0]), 2 )

mic1 = mic1.tolist()

fig, ax = room.plot()
ax.set_xlim([-1, 10])
ax.set_ylim([-1, 10]);


mic_locs = np.c_[
    mic1, mic4,
]


# finalmente posiciona  o array na sala
room.add_microphone_array(mic_locs)

# Executa a simulação (isso também construirá o RIR automaticamente)
room.simulate()

# mede o tempo de reverberação
rt60 = room.measure_rt60()
print("The desired RT60 was {}".format(rt60_tgt))
print("The measured RT60 is {}".format(rt60[1, 0]))

# Plot
plt.figure()

# plota a RIR - room.plot_rir()
rir_1_0 = room.rir[1][0]
rir_2_0 = room.rir[0][0]

plt.subplot(2, 1, 1)
plt.plot(np.arange(len(rir_1_0)) / room.fs, rir_1_0)
plt.title("The RIR from source 0 to mic 1")
plt.xlabel("Time [s]")

# plota o sinal no microfone 1
plt.subplot(2, 1, 2)
plt.plot(room.mic_array.signals[1, :])
plt.title("Microphone 1 signal")
plt.xlabel("Time [s]")
```

```
        plt.tight_layout()
        plt.show()
        plt.close()

        librosa.output.write_wav('Pyroom_mic1_Dim' + str(room_dim) + '_RT_' + str(round(rt60_tgt,2)) +
'_seg' +str(int(round(time.time(),0)))+'.wav', room.rir[1][0], room.fs, norm=False)
        librosa.output.write_wav('Pyroom_mic2_Dim' + str(room_dim) + '_RT_' + str(round(rt60_tgt,2)) +
'_seg' + str(int(round(time.time(),0)))+'.wav', room.rir[0][0], room.fs, norm=False)

def generate_RIR():
    i = 0;
    # i representa o número total de RIR geradas
    while (i < 30000):
        try:
            RIR_generator('Excitacao.wav')
        except Exception:
            # Em qualquer caso de erro durante a execução da função
            pass

        i = i +1
        print(i)


# Remova o comentário para executar o método generate_RIR() que
# gera as curvas RIR
# generate_RIR()
```

**2 - GERAÇÃO DAS AMOSTRAS DE RUÍDO DE FUNDO (BGN)**

### 3 - CÁLCULO DA DENSIDADE ESPECTRAL DE POTÊNCIA (PSD)

```
file_name = "/kaggle/input/sti-prediction/RVB2014_type1_rir_largeroom1_far_angla.wav"
sig, sr = librosa.load(file_name)

freqs, psd = signal.welch(sig,sr)

plt.figure(figsize=(5, 4))
plt.semilogx(freqs, psd)
plt.title('PSD: Densidade Espectral de Potência')
plt.xlabel('Frequência')
plt.ylabel('Potência')
plt.tight_layout()
plt.show()
plt.close()

N = len(sig)
T = 1 / sr
yf = fft(sig)
xf = np.linspace(0.0, 1.0/(2.0*T), N//2)
plt.plot(xf, 2.0/N * np.abs(yf[0:N//2]))
plt.grid()

plt.show()
plt.close()

s = Signal.from_wav(file_name)
s.fs
s.channels
s.samples

s.plot_power_spectrum()
s.spectrogram()
s.plot_levels()
s.plot_octaves()
fig = s.plot_third_octaves()

plt.show()
plt.close()

bands = acoustics.bands.octave(125, 2000)
b = acoustics.room.t60_impulse(file_name, bands, rt='t30')
print("Tempo de reverberação")
print(b)

print('Adição de ruído')

filename_BGN = "/kaggle/input/sti-prediction/noise-free-sound-0042.wav"
s2 = Signal.from_wav(filename_BGN)
BGN_octave = acoustics.signal.octaves(s2,
                                      s2.fs,
                                      density=False,
                                      frequencies=[ 63.,
                                                    125.,
                                                    250.,
                                                    500.,
                                                    1000.,
                                                    2000],
```

```
                                ref=2e-05)

RIR_level  = Signal.from_wav(file_name)

Speech_level = acoustics.signal.octaves(RIR_level,
                                RIR_level.fs,
                                density=False,
                                frequencies=[63.,
                                             125.,
                                             250.,
                                             500.,
                                             1000.,
                                             2000],
                                ref=2e-05)

print('Ruído de fundo')
print(BGN_octave)

print('Nível operacional da Fala')
print(Speech_level)

print("SNR")
SNR = Speech_level[1] - BGN_octave[1]
print(SNR)

plt.show()
plt.close()
```

**4 - CRIAÇÃO DO CONJUNTO DE DADOS STI (DOMÍNIOS DE ORIGEM E ALVO)**

```
"""
- Valores simulados de STI via Equação de Schroeder
"""

def BGN_audios(folder):

    """
     Converter recursivamente WAV em pasta de matrizes de espectrograma -
     pasta a ser convertida.
    """

    allFiles = []

    for root, dirs, files in os.walk(folder):
        allFiles += [os.path.join(root, f) for f in files
                     if f.endswith('.wav')]

    return allFiles

def RIR_audios(folder):

    allFiles = []

    for root, dirs, files in os.walk(folder):
        allFiles += [os.path.join(root, f) for f in files
                     if f.endswith('.wav')]

    return allFiles

def RIR_BGN_2_STI(RIR_file, BGN_file, id_number):

    sig_BGN, sr_BGN = librosa.load(BGN_file)
    freqs_BGN, psd_BGN = signal.welch(sig_BGN,sr_BGN)

    sig_RIR, sr_RIR = librosa.load(RIR_file)
    freqs_RIR, psd_RIR = signal.welch(sig_RIR, sr_RIR)

    # Assegura que avalia somente frequências menores que 8000 Hz
    psd_BGN = psd_BGN[0:len(freqs_BGN[freqs_BGN < 8000])]
    psd_RIR = psd_RIR[0:len(freqs_RIR[freqs_RIR< 8000])]

    # Interpola
    xvals = np.arange(0,8001,10)
    psd_BGN = np.interp(xvals, freqs_BGN[freqs_BGN < 8000], psd_BGN)
    psd_RIR = np.interp(xvals, freqs_RIR[freqs_RIR< 8000], psd_RIR)

    # Normaliza a potência
    psd_BGN = 10 * np.log10(psd_BGN)
    psd_RIR = 10 * np.log10(psd_RIR)


    RT_octave = acoustics.room.t60_impulse(RIR_file,
                                           acoustics.bands.octave(125, 8000),
                                           rt='t30')

    dummy_BGN = Signal.from_wav(BGN_file)
    BGN_octave = acoustics.signal.octaves(dummy_BGN,
```

```
                        dummy_BGN.fs,
                        density=False,
                        frequencies=acoustics.bands.octave(125, 8000),
                        ref=2e-05)

Ln = BGN_octave[1]

# conforme a norma IEC 60268-16
Op_SL_octave = [61.4, 65.6, 62.3, 56.8, 51.3, 42.6, 33.6]


modulation_freq = np.array([0.63,
                        0.8,
                        1.0,
                        1.25,
                        1.6,
                        2.0,
                        2.5,
                        3.15,
                        5.0,
                        6.3,
                        8.0,
                        10,
                        12.5])

octave_freq = np.array([125, 250, 500, 1000, 2000, 4000, 8000])

# Inicialização
mTF = np.zeros((len(modulation_freq), len(octave_freq)))
SNR = np.zeros((len(modulation_freq), len(octave_freq)))
TI = np.zeros((len(modulation_freq), len(octave_freq)))
MTI = np.zeros(len(octave_freq))

# filtros
alphas_Males = np.array([0.085, 0.127, 0.230, 0.233, 0.309, 0.224, 0.173])
betas_Males =  np.array([0.085, 0.078, 0.065, 0.011, 0.047, 0.095])

for m in range(0,mTF.shape[0]):

    for octave in range(0,mTF.shape[1]):

        mTF[m,octave] = (( 1 + (2*math.pi*modulation_freq[m]*RT_octave[octave]/ 13.82)**2 )**
                    (-1/2))*(1 + 10**( -(Op_SL_octave[octave]  - Ln[octave])/10 ))**(-1)

        SNR[m,octave] = 10 * np.log10( mTF[m,octave] / (1 - mTF[m,octave] ))

        if SNR[m,octave] > 15:
            SNR[m,octave] = 15
        if SNR[m,octave] < -15:
            SNR[m,octave] = -15

        TI[m,octave] = (SNR[m,octave] + 15 ) / 30

        MTI[octave] = ( 1 / len(modulation_freq) ) * sum( TI[:,octave])

        if MTI[octave] > 1:
            MTI[octave] = 1

        STI = np.dot(alphas_Males,MTI)
```

```
    room_id = id_number  * np.ones(( len(xvals ), 1) )
    room_STI = np.round(STI,3) * np.ones(( len(xvals ), 1) )

    output = np.concatenate([room_id, room_STI,
                        xvals.reshape(len(xvals),1),
                        psd_RIR.reshape(len(psd_RIR),1),
                        psd_BGN.reshape(len(psd_BGN),1) ] , 1)

    return  output

folder_RIR = r"RIR_simuladas"
folder_BGN = r"Background_Noise"

BGN = BGN_audios(folder_BGN)
RIR = RIR_audios(folder_RIR)

# Cria amostras
id_number = 0
dummy_RIR = random.randint(0,10500)
dummy_BGN = random.randint(0,686)

def generate_STI_database():

    A = RIR_BGN_2_STI(RIR[dummy_RIR], BGN[dummy_BGN], id_number)

    id_number = 1
    while id_number < 30001:

        dummy_RIR = random.randint(0,8)
        dummy_BGN = random.sample(range(1, 686), 5)

        for i in dummy_BGN:
            A = np.vstack([A, RIR_BGN_2_STI(RIR[dummy_RIR], BGN[i], id_number )])
            id_number =  id_number + 1

            # salva os dados
            # pd.DataFrame(A).to_csv("Train_Database_identification.csv")

        print(id_number)

    dataset = pd.DataFrame({'c1': A[:,0], 'c2': A[:,1],
                        'c3': A[:,2], 'c4': A[:,3],
                        'c5': A[:,4]})

    # dataset.to_pickle("./dummy.pkl")
    # savez_compressed('data_compres.npz', A)
    # unpickled_df = pd.read_pickle("./dummy.pkl")
```

## 5 - TREINAMENTO DA REDE AUTOENCODER VARIACIONAL (VAE) NO DOMÍNIO DE ORIGEM

```
# Faz a carga dos dados gerados na seção 4 do código

A = np.load('../input/sti-prediction/data_compres.npz')
A.files
A = A['arr_0']

df = pd.DataFrame({'c1': A[:,0], 'c2': A[:,1],
                   'c3': A[:,2], 'c4': A[:,3],
                   'c5': A[:,4]})

"""
2 - Configurações de funções
"""

def feature_normalize(dataset):

    mu = np.mean(dataset, axis=0)
    sigma = np.std(dataset, axis=0)
    return (dataset - mu)/sigma


def show_confusion_matrix(validations, predictions):

    matrix = metrics.confusion_matrix(validations, predictions)
    plt.figure(figsize=(6, 4))
    sns.heatmap(matrix,
                cmap="coolwarm",
                linecolor='white',
                linewidths=1,
                xticklabels=LABELS,
                yticklabels=LABELS,
                annot=True,
                fmt="d")

    plt.title("Confusion Matrix")
    plt.ylabel("True Label")
    plt.xlabel("Predicted Label")
    plt.show()
    plt.close()


def show_basic_dataframe_info(dataframe, preview_rows=20):

    """
    Esta função mostra informações básicas para o dataframe fornecido
    Argumentos:
        dataframe: Um DataFrame do Pandas que deve conter dados
        preview_rows: um valor inteiro de quantas linhas visualizar
     Retorna:
         Nada
    """

    # Informa quantas linhas e colunas contém nos dados de treinamento
    print("Número de colunas no dataframe: %i" % (dataframe.shape[1]))
    print("Número de linhas no dataframe: %i\n" % (dataframe.shape[0]))
    print("Printa as primeiras 20 linhas")
```

```
    print(dataframe.head(preview_rows))
    print("\nDescrição dataframe:\n")


def read_data(file_path):

    """
    Esta função é o processo ETL dos valores STI
    Argumentos:
        file_path: URL apontando para o arquivo pickle
    Retorna:
        Um dataframe do pandas
    """

    df.rename(columns = {'c1': 'Room-id',
                         'c2': 'STI',
                         'c3': 'Freq - [Hz]',
                         'c4': 'PSD(RIR)',
                         'c5': 'PSD(BGN)' },
                         inplace=True)

    # Isso é muito importante, caso contrário o modelo não terá o "fit" e a função custo
    # aparecerá como NAN

    label_encoder = LabelEncoder()

    # número de classes associadas aos valores do STI
    n_bins = 10
    y = label_encoder.fit_transform(pd.cut(df['STI'],
                                           n_bins,
                                           retbins=True)[0]
                                    )

    df['STI'] = y
    df.dropna(axis=0, how='any', inplace=True)

    return df


def convert_to_float(x):

    try:
        return np.float(x)
    except:
        return np.nan


def feature_normalize(dataset):

    mu = np.mean(dataset, axis=0)
    sigma = np.std(dataset, axis=0)
    return (dataset - mu)/sigma


def plot_axis(ax, x, y, title):

    ax.plot(x, y)
    ax.set_title(title)
    ax.xaxis.set_visible(False)
```

```
    ax.set_ylim([min(y) - np.std(y), max(y) + np.std(y)])
    ax.set_xlim([min(x), max(x)])
    ax.grid(True)


def plot_activity(activity, data):

    fig, (ax0, ax1) = plt.subplots(nrows=2,
        figsize=(15, 10),
        sharex=True)
    plot_axis(ax0, data['Freq - [Hz]'], data['PSD(RIR)'], 'PSD(RIR)')
    plot_axis(ax1, data['Freq - [Hz]'], data['PSD(BGN)'], 'PSD(BGN)')
    plt.subplots_adjust(hspace=0.2)
    fig.suptitle(activity)
    plt.subplots_adjust(top=0.90)
    plt.show()
    plt.close()


def create_segments_and_labels(df, freq_steps, step, label_name):


    """
     Esta função recebe um dataframe e retorna os segmentos remodelados
     de dados BGN, RIR, bem como os rótulos STI correspondentes
     Argumentos:
        df: Dataframe no formato esperado
        freq_steps: valor inteiro do comprimento de um segmento criado
     Retorna:
        remodelados_segmentos
        rótulos:
     """

    # indica duas variáveis sob análise PSD(RIR) e PSD(BGN)
    N_FEATURES = 2

    # Número de passos para avançar em cada iteração (deve ser sempre
    # igual a freq_steps para não haver sobreposição entre os segmentos)
    # passo = freq_steps

    segments = []
    labels = []

    for i in range(0, len(df) - freq_steps, step):
        xs = df['PSD(RIR)'].values[i: i + freq_steps]
        ys = df['PSD(BGN)'].values[i: i + freq_steps]
        xs = xs[1:]
        ys = ys[1:]

        # Recupera o rótulo usado com mais frequência neste segmento
        label = stats.mode(df[label_name][i: i + freq_steps])[0][0]
        segments.append([xs, ys])
        labels.append(label)

    # Ajusta o forma dimensional dos segmentos para uma melhor representação visual
    reshaped_segments = np.asarray(segments,
                                   dtype= np.float32).reshape(-1,
                                                              freq_steps-1,
                                                              N_FEATURES)
```

```
    labels = np.asarray(labels)

    return reshaped_segments, labels

"""
## Criar banco de dados de treinamento
"""

pd.options.display.float_format = '{:.3f}'.format

print('keras version ', keras.__version__)

LABELS = ["1","2","3","4","5","6","7","8","9","10"]
FREQ_PERIODS = len(np.arange(0,8001,10))
STEP_DISTANCE = FREQ_PERIODS

print("\n--- Carregar, inspecionar e transformar dados ---\n")
print("Carregar conjunto de dados do npz")

read_data(df)

show_basic_dataframe_info(df, 20)

print("\n--- Remodele os dados em segmentos ---\n")

# Diferencia entre conjunto de teste e conjunto de treinamento
df_train = df[df['Room-id'] <= 25000]
df_test = df[df['Room-id'] > 25000]

# Normaliza recursos para conjunto de dados de treinamento
scaler = MinMaxScaler()
column_names_to_normalize = ['PSD(RIR)','PSD(BGN)']

x = df_train[column_names_to_normalize].values
x_scaled = scaler.fit_transform(x)

df_temp = pd.DataFrame(x_scaled,
                       columns=column_names_to_normalize,
                       index=df_train.index)

df_train[column_names_to_normalize] = df_temp

# Para o grupo de testes
x = df_test[column_names_to_normalize].values
x_scaled = scaler.fit_transform(x)
df_temp = pd.DataFrame(x_scaled,
                       columns=column_names_to_normalize,
                       index=df_test.index)

df_test[column_names_to_normalize] = df_temp

# Arredondar
df_train = df_train.round({'PSD(RIR)': 6, 'PSD(BGN)': 6})
df_test =  df_test.round({'PSD(RIR)': 6, 'PSD(BGN)': 6})

# Remodele os dados de treinamento em segmentos, para que
# eles possam ser processados pela rede VAE
# Define o nome da coluna do vetor de rótulo
```

```
LABEL = "STI"

x_train, y_train = create_segments_and_labels(df_train,
                                              FREQ_PERIODS,
                                              STEP_DISTANCE,
                                              LABEL)


x_test, y_test = create_segments_and_labels(df_test,
                                            FREQ_PERIODS,
                                            STEP_DISTANCE,
                                            LABEL)

plt.show()
plt.close()

print("\n--- Remodelar dados para serem aceitos pelo módulo Keras ---\n")

# Inspeciona x data
print('x_train shape: ', x_train.shape)
print(x_train.shape[0], 'amostras de treinamento')

# Inspeciona y data
print('y_train shape: ', y_train.shape)
print('y_test shape: ', y_test.shape)

# Verifica input_shape / reshape para o módulo Keras
print("Verificando as dimensões do conjunto de dados de treinamento")

print("Dimensões antes")
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

# Dimensão de entrada
max_psd_freq = 1600

# Os canais são os últimos no back-end do TensorFlow
print("Dimensão após reestruturação - treino")
x_train_reshaped = x_train.reshape(x_train.shape[0], max_psd_freq)
print(x_train_reshaped.shape)

# Agora verifica para os dados de teste
print("Dimensão após reestruturação - teste")
x_test_reshaped = x_test.reshape(x_test.shape[0], max_psd_freq)
print(x_test_reshaped.shape)

x_tr = x_train_reshaped
x_te = x_test_reshaped
plt.close()
```

### 5.1 - ESTATÍSTICAS DE TREINAMENTO PARA O DOMÍNIO DE ORIGEM

```
# Fitting - Extrai a coluna do dataframe
# e converta-o em um array numpy

data_source_ANOVA = y_train/10

# Ajustar para uma distribuição gaussiana aos dados
mu, std = stats.norm.fit(data_source_ANOVA)

# Plot
plt.hist(data_source_ANOVA, bins=20, alpha=0.6, color='r')
plt.xlabel('STI')
plt.ylabel('Frequência')

print(mu, std)

plt.show()
plt.savefig('Figura_21_a.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
# Plota o histograma dos dados de origem
plt.hist(data_source_ANOVA, bins=20, density=True, alpha=0.6, color='r')

# Plota o ajuste normal para os dados de origem
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, mu, std)

plt.xlabel('STI')
plt.ylabel('Função densidade de probabilidade')

ax2 = plt.twinx()
ax2.plot(x, p, 'k--')
ax2.set_ylabel('Ajuste distribuição normal', color='k')

plt.show()
print(mu, std)

plt.savefig('Figura_21_c.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
plt.hist(data_source_ANOVA, bins=20, density=True, alpha=0.6, color='r')

xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
c = stats.norm.cdf(x, mu, std)

plt.xlabel('STI')
plt.ylabel('Função densidade de probabilidade')
ax2 = plt.twinx()

ax2.plot(x, c, 'k--')
ax2.set_ylabel('Função densidade acumulada ajustada normal', color='k')

print(mu, std)
plt.savefig('Figura_21_e.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
plt.show()
plt.close()
```

**5.2 - TREINAMENTO DA REDE AUTOENCODER VARIACIONAL (VAE)**

```
""" ## 3 - Redimensionando os dados para o treinamento VAE """

print(" Modelos VAE e define seus parâmetros ")

batch_size, n_epoch = 1, 10

# Aqui muda a dimensão z (espaço latente)
n_hidden, z_dim = 64, 2

# encoder
x = Input(shape=(x_tr.shape[1:]))
x_encoded = Dense(n_hidden, activation='relu')(x)
x_encoded = Dense(n_hidden//2, activation='relu')(x_encoded)
mu = Dense(z_dim)(x_encoded)
log_var = Dense(z_dim)(x_encoded)

# função de amostragem
def sampling(args):
    mu, log_var = args
    eps = K.random_normal(shape=(batch_size, z_dim), mean=0., stddev=1.0)
    return mu + K.exp(log_var) * eps

z = Lambda(sampling, output_shape=(z_dim,))([mu, log_var])

# decoder
z_decoder1 = Dense(n_hidden//2, activation='relu')
z_decoder2 = Dense(n_hidden, activation='relu')
y_decoder = Dense(x_tr.shape[1], activation='sigmoid')

z_decoded = z_decoder1(z)
z_decoded = z_decoder2(z_decoded)
y = y_decoder(z_decoded)

# loss
reconstruction_loss = objectives.binary_crossentropy(x, y) * x_tr.shape[1]
kl_loss = 0.5 * K.sum(K.square(mu) + K.exp(log_var) - log_var - 1, axis = -1)
vae_loss = reconstruction_loss + kl_loss

# Monta o modelo da rede VAE: codificador + decodificador
vae = Model(x, y)
vae.add_loss(vae_loss)

# Adiciona métricas de perda do VAE
vae.add_metric(reconstruction_loss, name='reconstruction_loss')
vae.add_metric(kl_loss, name='kl_loss')
vae.add_metric(vae_loss, name='vae_loss')
vae.compile(optimizer='rmsprop')
vae.summary()

vae.fit(x_tr,
        shuffle=True,
        epochs=n_epoch,
        batch_size=batch_size,
        validation_data=(x_te, None), verbose=1)

# Modelo codificador
encoder = Model(x, mu)
encoder.summary()
```

### 5.3 - VALORES DA FUNÇÃO PERDA DA REDE VAE

```
print(vae.history)
print(vae.history.history.keys())

loss_values = vae.history.history['loss']

plt.plot(loss_values)
plt.xlabel('Época')
plt.ylabel('Função de perda')
plt.show()
plt.close()

# Lista de todas as métricas possíveis para plotar:
# Extrai os valores de perdas do histórico

component1_values = vae.history.history['reconstruction_loss']
component2_values = vae.history.history['val_reconstruction_loss']

plt.plot(component1_values, 'go-', label='Função perda de reconstrução')
plt.plot(component2_values, label='Validação - Função perda de reconstrução')
plt.xlabel('Época')
plt.ylabel('Função de perda')
plt.legend()
plt.show()
plt.savefig('Figura_24_a.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()

# Extrai os valores de perda KL do histórico e plota

component1_values = vae.history.history['kl_loss']
component2_values = vae.history.history['val_kl_loss']

plt.plot(component1_values, 'g--', label='Função perda de KL')
plt.plot(component2_values, label='Validação - Função perda de KL')

plt.xlabel('Época')
plt.ylabel('Função de perda')
plt.legend()
plt.show()
plt.savefig('Figura_24_b.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
component1_values = vae.history.history['vae_loss']
component2_values = vae.history.history['val_vae_loss']

plt.plot(component1_values, 'g--', label='VAE - Função perda total')
plt.plot(component2_values, label='Validação VAE - Função perda total')

plt.xlabel('Época')
plt.ylabel('Função de perda')
plt.legend()
plt.show()
plt.savefig('Figura_24_c.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
```

## 6- OTIMIZAÇÃO DOS HIPERPARÂMETROS (HPO) DO MODELO VAE

```
def start_Autoencoder(features,trials,  plot_graph = False):

 # Modelo VAE
 def create_model(func_ativacao, num_neuronios, zdim_num, kernel_initializer):
  clear_session()

  batch_size, n_epoch = 1, 1
  n_hidden, z_dim = num_neuronios, zdim_num

  # encoder
  x = Input(shape=(x_tr.shape[1:]))
  x_encoded = Dense(n_hidden,
                    activation=func_ativacao,
                    kernel_initializer=kernel_initializer)(x)
  x_encoded = Dense(n_hidden//2, activation=func_ativacao)(x_encoded)

  mu = Dense(z_dim)(x_encoded)
  log_var = Dense(z_dim)(x_encoded)

  # função de amostragem
  def sampling(args):
     mu, log_var = args
     eps = K.random_normal(shape=(batch_size, z_dim), mean=0., stddev=1.0)
     return mu + K.exp(log_var) * eps

  z = Lambda(sampling, output_shape=(z_dim,))([mu, log_var])

  # decoder
  z_decoder1 = Dense(n_hidden//2, activation='relu')
  z_decoder2 = Dense(n_hidden, activation='relu')
  y_decoder = Dense(x_tr.shape[1], activation='sigmoid')

  z_decoded = z_decoder1(z)
  z_decoded = z_decoder2(z_decoded)
  y = y_decoder(z_decoded)

  # função de perda
  reconstruction_loss = objectives.binary_crossentropy(x, y) * x_tr.shape[1]
  kl_loss = 0.5 * K.sum(K.square(mu) + K.exp(log_var) - log_var - 1, axis = -1)
  vae_loss = reconstruction_loss + kl_loss

  # Monta o modelo VAE: codificador + decodificador
  vae = Model(x, y)

  vae.add_loss(vae_loss)
  # Adiciona métricas de perda do VAE
  vae.add_metric(reconstruction_loss, name='reconstruction_loss')
  vae.add_metric(kl_loss, name='kl_loss')
  vae.add_metric(vae_loss, name='vae_loss')
  autoencoder = vae

  return autoencoder
```

```
# Função objetivo para otimizar pelo módulo OPTUNA

def objective(trial):
  func_ativacao = trial.suggest_categorical("func_ativacao",
                                  ["relu", "sigmoid", "swish"])
  num_neuronios = trial.suggest_int("num_neuronios", 2,64)
  zdim_num = trial.suggest_int("dim_espaco_latente", 2,128)

  if (func_ativacao == "relu"):
    model = create_model(func_ativacao,
                         num_neuronios,
                         zdim_num,
                         kernel_initializer="HeUniform")
  else:
    model = create_model(func_ativacao,
                         num_neuronios,
                         zdim_num,
                         kernel_initializer="GlorotUniform")

  model.compile(optimizer='rmsprop')


  # Implementar critério de parada antecipada.
  # O processo de treinamento para quando não há melhoria durante 50 iterações

  callback = keras.callbacks.EarlyStopping(monitor='loss', patience=50)
  history = model.fit(features,
                      features,
                      batch_size = 1,
                      epochs=10,
                      callbacks = [callback],
                      verbose = 0)

  return history.history["loss"][-1]

study = optuna.create_study(direction='minimize')

study.optimize(objective, n_trials=trials)

# Cria o modelo final com os melhores hiperparâmetros

print('Melhores hiperparâmetros encontrados via Optuna: \n', study.best_params)
if (study.best_params['func_ativacao'] == "relu"):
  model = create_model(study.best_params['func_ativacao'],
                       int(study.best_params['num_neuronios']),
                       int(study.best_params['dim_espaco_latente']),
                       kernel_initializer="HeUniform")
else:
  model = create_model(study.best_params['func_ativacao'],
                      int(study.best_params['num_neuronios']),
                      int(study.best_params['dim_espaco_latente']),
                      kernel_initializer="GlorotUniform")

model.compile(optimizer='rmsprop')
model.summary()
```

```
    # Implementar critério de parada antecipada.
    # O processo de treinamento para quando não há melhoria durante um certo número de iterações

    callback = keras.callbacks.EarlyStopping(monitor='loss', patience=50)
    history = model.fit(features,
                        features,
                        batch_size = 1,
                        epochs=2,
                        callbacks = [callback],
                        verbose = 0)

    result = model.predict(features)

    # Avaliação de resultados

    print(f'RMSE Autoencoder: {np.sqrt(mean_squared_error(features, result))}')
    print('')

    # Os seguintes valores são retornados: extract_f || MSE || Melhores hiperparâmetros OPTUNA

    return mean_squared_error(features, result), study.best_params, study


# Execute a otimização de hiperparâmetros HPO no modelo VAE

AutoEncoder_MSE, AutoEncoder_hyperparams, Study = start_Autoencoder(features = x_tr,
                                trials = 10,
                                plot_graph=True)
df_VAE = Study.trials_dataframe()
df_VAE
plot_optimization_history(Study)
plot_param_importances(Study)
optuna.visualization.plot_param_importances(
    Study, target=lambda t: t.duration.total_seconds(), target_name="duration"
)
plot_edf(Study)
plot_contour(Study, target_name="Função objetivo")

# Obtém os parâmetros classificados pelos valores de importância

importances = optuna.importance.get_param_importances(Study)
params_sorted = list(importances.keys())
params_sorted
importances
AutoEncoder_hyperparams
```

**7 - VISUALIZAÇÃO DOS EMBEDDINGS GERADOS VIA VAE**

```
cmap = matplotlib.cm.get_cmap('Spectral')

# Define a correspondência entre os rótulos
label_mapping = {1: 0.1, 2: 0.2, 3: 0.3, 4: 0.4, 5: 0.5, 6: 0.6, 7: 0.7, 8: 0.8, 9: 0.9}

# Função para plotar a representação
def plot_representation(features, labels, rep_type, pca_variance_ratio):
    fig = plt.figure(figsize=(15, 6))
    gs = GridSpec(2, 4, width_ratios=[1, 1, 1, 0.2])

    axes = [[None, None, None], [None, None, None]]
    for i, pair in enumerate(combinations([0, 1, 2, 3], 2)):
        row, col = i // 3, i % 3
        axes[row][col] = plt.subplot(gs[row, col])
        for label, color in zip([1, 2, 3, 4, 5, 6, 7, 8, 9], ['blue', 'green', 'black', 'yellow', 'brown', 'orange',
'red', 'gray', 'olive', 'gold']):
            new_label = label_mapping[label]
            axes[row][col].scatter(features[labels == label, pair[0]],
                                   features[labels == label, pair[1]],
                                   c=color,
                                   alpha=0.5,
                                   label=new_label)
        axes[row][col].set_xlabel(f'Dimensão {pair[0]+1}')
        axes[row][col].set_ylabel(f'Dimensão {pair[1]+1}')

    cax = plt.subplot(gs[:, 3])
    handles, labels = axes[0][0].get_legend_handles_labels()
    cax.legend(handles, labels, title='STI')
    cax.set_axis_off()


    plt.tight_layout(rect=[0, 0, 0.9, 0.95])
    plt.show()
    plt.savefig('Figura_26.png', dpi=300, bbox_inches='tight', transparent=True)
    plt.close()


    plt.figure(figsize=(8, 4))
    plt.bar(range(1, len(pca_variance_ratio) + 1), pca_variance_ratio, alpha=0.5, align='center')
    plt.xlabel('Componente Principal')
    plt.ylabel('Variância Explicada')
    plt.show()
    plt.savefig('Figura_27_a.png', dpi=300, bbox_inches='tight', transparent=True)
    plt.close()


    cumulative_variance = np.cumsum(pca_variance_ratio)
    plt.figure(figsize=(8, 4))
    plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance, marker='o', linestyle='-')
    plt.xlabel('Número de Componentes Principais')
    plt.ylabel('Variância Cumulativa Explicada')
    plt.grid(False)
    plt.show()
    plt.savefig('Figura_27_b.png', dpi=300, bbox_inches='tight', transparent=True)
    plt.close()
```

```
# Aplica PCA aos dados
PCA_transformer = PCA(n_components=4)
x_train_reshaped_TSE = np.asarray(x_train_reshaped, dtype='float64')
PCA_representation = PCA_transformer.fit_transform(x_train_reshaped_TSE)

# Visualizar os resultados da redução dimensional
plot_representation(PCA_representation, y_train, 'PCA', PCA_transformer.explained_variance_ratio_)

# Variância explicada por componente principal
explained_variance = PCA_transformer.explained_variance_ratio_

# Variância cumulativa explicada
cumulative_variance = np.cumsum(explained_variance)

# Número do componente principal
component_number = range(1, len(explained_variance) + 1)

# Cria um DataFrame
pca_info_df = pd.DataFrame({
    'Componente Principal': component_number,
    'Variância Explicada': explained_variance,
    'Variância Cumulativa Explicada': cumulative_variance
})

# Exibir o DataFrame
print(pca_info_df)

# Obter os componentes principais
principal_components = PCA_transformer.components_
```

## 8 - SALVANDO O MODELO CODIFICADOR DO VAE NUMA EXTENSÃO DE ARQUIVO .H5

```
# Salvando as matrizes de pesos

# vae.save_weights('meu_vae_weights.h5')
# codificador.save("meu_h5_model.h5")

# Carrega os pesos
# `save('codificador_VAE_model.h5')` cria um arquivo h5 `codificador_VAE_model.h5`.
# codificador.save("meu_h5_model.h5")

# Pode ser usado para reconstruir o modelo de forma idêntica.
# reconstructed_model = keras.models.load_model("meu_h5_model.h5")

# Vamos verificar: (Teste para verificar se o modelo foi recarregado corretamente)
# np.testing.assert_allclose(
# codificador.predict(test_input), reconstructed_model.predict(test_input)
# )

# O modelo reconstruído já está compilado e manteve o otimizador
# estado, para que o treinamento possa ser retomado:
# reconstruído_model.fit(test_input, test_target)
# Executando `save('codificador_VAE_model.h5')` cria um arquivo h5 `codificador_VAE_model.h5`.

encoder.save("vae_encoder.h5")

# Pode ser usado para reconstruir o modelo de forma idêntica.
# Carregue o modelo salvo e faça previsões

reconstructed_model = keras.models.load_model("vae_encoder.h5")
x_te_latent_recons = reconstructed_model.predict(x_te, batch_size=1)

# Compare com o modelo de codificador original
x_te_latent_origin = encoder.predict(x_te, batch_size=batch_size)

# Campara os resultados da reconstrução
x_te_latent_origin == x_te_latent_recons
```

## 8.1 - SALVANDO O MODELO DO CODIFICADOR COMO UM OBJETO GRÁFICO DO TENSORFLOW

```
encoder.save("codificador_VAE_model.model")

# Carrega o modelo
tensorflow_graph = tf.saved_model.load("./codificador_VAE_model.model")

# Preparando para realizar inferências
z_latent = tensorflow_graph(x_te ,False, None).numpy()

# x = np.random.uniform(size=(1600))
# x = np.expand_dims(x, axis=0)
# predicted = tensorflow_graph(x ,False, None).numpy()

# Compara os resultados do modelo salvo e do modelo original na memória
np.round(z_latent,4) == np.round(x_te_latent_origin,4)
```

## 8.2 - SALVANDO AS PREVISÕES DO CONJUNTO DE DADOS DE TREINAMENTO (X_TR_LATENT) NO ESPAÇO CODIFICADO PARA MODELAGEM KPCR

```
# Importante: Gere a codificação latente z com 128 dimensões, então z = 2
# x_tr

x_tr_latent = encoder.predict(x_tr, batch_size=batch_size)

# Faça um dataframe do pandas do espaço latente e exporte os dados

x_tr_latent.shape

# Criando o dataframe do pandas

dataset_high_z_dim = pd.DataFrame(x_tr_latent)
classes = pd.DataFrame(y_train)

# dataset_high_z_dim.append(classes)
dataset_high_z_dim['classes'] = classes
print(dataset_high_z_dim)

dataset_high_z_dim.to_csv('dataset_high_z_dim.csv')
```

## 9 - PLOTANDO A VISUALIZAÇÃO DO ESPAÇO KERNEL 2D

```
# Inspecionando as variáveis do espaço latente (z1 e z2)
x_tr_latent
```

## 9.1 - PLOTANDO A REPRESENTAÇÃO DO ESPAÇO LATENTE (Z = 2) DO TREINAMENTO (CONJUNTO DE DADOS DE ORIGEM)

```
# Plota
plt.figure(figsize=(6, 6))
y_tr = y_train
plt.scatter(x_tr_latent[:, 0], x_tr_latent[:, 1], c=y_tr/10)
plt.colorbar()

plt.legend(title="STI - Dados de treinamento (origem) ")
plt.xlabel('Dimensão - z1')
plt.ylabel('Dimensão - z2')

plt.show()
plt.savefig('Figura_28_a.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
# Assumindo que y_tr contém os rótulos da classe e x_tr_latent contém os dados
# Calcular a variância para cada classe

unique_labels = np.unique(y_tr)

variances = []
for label in unique_labels:
    mask = (y_tr / 10) == label
    data_for_label = x_tr_latent[mask]
    variance_for_label = np.var(data_for_label)
    variances.append(variance_for_label)


plt.figure(figsize=(8, 6))
plt.bar(unique_labels, variances, color='blue', alpha=0.7)
plt.xlabel('Rótulo da classe')
plt.ylabel('Variância')
plt.title('Variação de dados por rótulo de classe')
plt.xticks(unique_labels)
plt.show()
plt.close()
```

## 9.2 - PLOTANDO A REPRESENTAÇÃO DO ESPAÇO LATENTE (Z = 2) DE TESTE (CONJUNTO DE DADOS DE ORIGEM)

```
# Faça a previsão do espaço latente
x_te_latent = encoder.predict(x_te, batch_size=batch_size)
y_te = y_test
plt.figure(figsize=(6, 6))
plt.scatter(x_te_latent[:, 0], x_te_latent[:, 1], c = y_te/10 )
plt.legend(title="STI - Dados de validação (domínio) ")
plt.xlabel('Dimensão - z1')
plt.ylabel('Dimensão - z2')
plt.colorbar()
```

```
plt.show()
plt.savefig('Figura_28_b.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
```

### 9.3 - RECONSTRUINDO O ESPAÇO LATENTE VAE VIA CODIFICADOR DE MAPAS IMBUÍDOS (DOMÍNIO DE ORIGEM)

```
x_tr_latent = encoder.predict(x_tr, batch_size=batch_size)
y_tr = y_train
dataset = pd.DataFrame({'z1': x_tr_latent[:, 0], 'z2': x_tr_latent[:, 1], 'STI':  y_tr})

print(dataset)
```

#### 9.3.1 - Visualização da inferência do histograma do espaço latente

```
fig, axes = plt.subplots(2, 3, figsize=(13, 9))

for i, col in enumerate(['z1', 'STI', 'z2']):
    dataset[col].plot(kind="hist", bins=20, density=True, alpha=0.6, ax=axes[0, i], label=col, color='blue')

axes[0, 0].set_xlabel('Dimensão latente - z1')
axes[0, 0].set_ylabel('Função densidade de probabilidade - KDE')
axes[0, 1].set_xlabel('STI')
axes[0, 1].set_ylabel('Função densidade de probabilidade - KDE')
axes[0, 2].set_xlabel('Dimensão latente - z2')
axes[0, 2].set_ylabel('Função densidade de probabilidade - KDE')


for i, col in enumerate(['z1', 'STI', 'z2']):
    data = dataset[col]
    mu, std = stats.norm.fit(data)

    sm.qqplot(data, line='s', ax=axes[1, i], color='blue')
    axes[1, i].set_title(f'{col} - Fit: $\mu={mu:.2f}$, $\sigma={std:.2f}$')

    axes[1, i].set_xlabel('Quantis teóricos (normal)')
    axes[1, i].set_ylabel('Quantis amostrais')

plt.tight_layout()
plt.show()
plt.savefig('Figura_34.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
```

### 9.3.2 - IA Explicativa (EAX) - Plotando a reconstrução do modelo codificador em 2 dimensões

```
# Definir dimensões
dim1 = 40
dim2 = 40

# dimensão da malha
n = 20

# Dimensão do espaço latente z
z_dim = 2

# Cria o modelo decodificador

decoder_input = Input(shape=(z_dim,))
_z_decoded = z_decoder1(decoder_input)
_z_decoded = z_decoder2(_z_decoded)
_y = y_decoder(_z_decoded)
generator = Model(decoder_input, _y)
grid_x = norm.ppf(np.linspace(0.05, 0.95, n))
grid_y = norm.ppf(np.linspace(0.05, 0.95, n))

figure = np.zeros((dim1 * n, dim2 * n))

for i, xi in enumerate(grid_x):
    for j, yi in enumerate(grid_y):
        z_sample = np.array([[xi, yi]])
        x_decoded = generator.predict(z_sample)
        image = x_decoded[0].reshape(dim1, dim2)
        figure[i * dim1: (i + 1) * dim1, j * dim2: (j + 1) * dim2] = image



plt.figure(figsize=(10, 10))
plt.xlabel('Dimensão latente - z1')
plt.ylabel('Dimensão latente - z2')
plt.imshow(figure, cmap='jet', extent=[0, 1, 0, 1])
cbar = plt.colorbar(fraction=0.046, pad=0.04)
cbar.set_label('STI')
plt.show()

plt.savefig('Figura_33.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
```

### 9.3.3 - PCA aplicado ao espaço latente durante o treinamento da rede VAE: z Dim = 2

```
print(x_te_latent_recons.shape)

print(y_te.min())
```

### 9.3.4 - Visualização da estimativa de densidade do kernel do domínio de origem

```
for sti_value in range(1, 10):
    ax =  dataset[dataset['STI'] == sti_value][['z1', 'z2']].plot.kde()
    plt.xlabel('Dimensão latente - z')
    plt.ylabel('Densidade de Kernel')
    plt.text(0.03, 0.9, f'STI = {sti_value / 10}', transform=plt.gca().transAxes, fontsize=12, color='black')
    plt.show()
    plt.savefig(f'Figura_29_origem_{sti_value / 10}.png', dpi=300, bbox_inches='tight',
transparent=True)
    plt.close()
```

### 9.3.5 - Mapeamento do conjunto de origem

```
z_mapping_VAE_source_domain = dataset
z_mapping_VAE_source_domain
```

### 9.3.6 - Visualização 2D do KDE do espaço codificado bidimensional e valores STI para os domínios de origem-destino

```
def plot_kde_for_sti(sti_value):

    plt.figure(figsize=(6, 6))

    x = dataset[dataset['STI'] == sti_value]['z1']
    y = dataset[dataset['STI'] == sti_value]['z2']

    nbins = 300
    k = kde.gaussian_kde([x, y])
    xi, yi = np.mgrid[x.min():x.max():nbins*1j, y.min():y.max():nbins*1j]
    zi = k(np.vstack([xi.flatten(), yi.flatten()]))

    plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='auto')
    plt.xlabel('Dimensão latente - z1')
    plt.ylabel('Dimensão latente - z2')
    plt.title(f'STI = {sti_value/10}')
    plt.show()
    plt.savefig(f'Figura_32_origem_{sti_value / 10}.png', dpi=300, bbox_inches='tight',
transparent=True)
    plt.close()


for sti_value in [1, 3, 6, 9]:
    plot_kde_for_sti(sti_value)
```

## 10 - IMPLEMENTAÇÃO DA REGRESSÃO EM COMPONENTES PRINCIPAIS VIA NÚCLEO - KPCR

### 10.1 - MODELAGEM DA REGRESSÃO DE KERNEL (KPCR)

```
# Carrega em memória o banco de dados
df_KPCR  = pd.read_csv('/kaggle/input/sti-prediction/dataset_high_z_dim_to_KPCR.csv')
df_KPCR.dropna(inplace=True)

# centralize os valores em cada coluna do DataFrame
df_KPCR = df_KPCR.groupby(by=["classes"], dropna=True).mean()
df_KPCR = df_KPCR.drop(columns=["Unnamed: 0"])

# Centralização pela média = 0
df_KPCR = df_KPCR.apply(lambda x: x-x.mean())

X_KPCR = df_KPCR.values
y_KPCR = df_KPCR.index.values

ax = df_KPCR.T.plot.kde(bw_method=0.3)
ax.set_xlim([-.2, 0.2])
plt.show()
plt.close()

# Obtém o kernel e calcula a respectiva transformação
kernel = RBF(0.2)
K = kernel.__call__(X_KPCR)

# Centraliza o kernel
eigenValues, eigenVectors = LA.eig(K)

# Verifica se o kernel é ortonormal
Z = K  * eigenVectors

gamma = LA.inv(np.transpose(Z) * Z) * np.transpose(Z) * y_KPCR
y_pred = np.diag(Z*gamma)

plt.scatter(y_KPCR, y_pred)
plt.xlabel('Dimensão latente - z1')
plt.ylabel('Dimensão latente - z2')
plt.legend(title="y = KPCR(X): Curva de Calibração")
plt.show()
plt.close()

# Fazendo o modelo prever a função
x_hat = X_KPCR[5]
y_hat = (np.transpose(np.expand_dims(y_KPCR, 1)) *  K \
        * kernel.__call__(X_KPCR, np.transpose(np.expand_dims(x_hat, 1)))).max()
```

### 10.2 - TESTE KS SOBRE A REGRESSÃO (KPCR)

```
# Supondo que já tenha o DataFrame df_KPCR com as colunas especificadas
# Gera gráficos do KDE para cada classe única

ax = df_KPCR.T.plot.kde(bw_method=0.3, legend=False)
ax.set_xlim([-.2, 0.2])
plt.title('KDE Plots for Unique Classes')
plt.xlabel('Values')
plt.ylabel('Density')

# Calcula a estatística de teste KS para cada par de gráficos do KDE
ks_results = []
unique_classes = df_KPCR.index.unique()

for i, class1 in enumerate(unique_classes):
    for j, class2 in enumerate(unique_classes):
        if i < j:
            kde1 = df_KPCR[df_KPCR.index == class1].values.flatten()
            kde2 = df_KPCR[df_KPCR.index == class2].values.flatten()

            ks_stat, p_value = ks_2samp(kde1, kde2)

            ks_results.append({
                'Class 1': class1/10,
                'Class 2': class2/10,
                'KS Test Statistic': ks_stat,
                'P-Value': p_value,
            })

# Cria um DataFrame para exibir os resultados do teste KS
ks_df = pd.DataFrame(ks_results)

plt.figure(figsize=(10, 4))
plt.hist(ks_df['P-Value'], bins=20, color='skyblue', alpha=0.7)
plt.title('P-Value Histogram')
plt.xlabel('P-Value')
plt.ylabel('Frequency')

plt.figure(figsize=(10, 6))
heatmap_data = ks_df.pivot(index='Class 1', columns='Class 2', values='KS Test Statistic')
sns.heatmap(heatmap_data, cmap='coolwarm', annot=True, fmt=".3f", cbar=True)
plt.xlabel('STI')
plt.ylabel('STI')


plt.legend(unique_classes)
plt.show()

plt.savefig('Figura_31.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
ks_df
# Gera o ks_df DataFrame para criar uma matriz quadrada
ks_matrix = ks_df.pivot_table(index='Class 1', columns='Class 2', values='KS Test Statistic',
fill_value=0)

print(ks_matrix)
# Gera o ks_df DataFrame para criar uma matriz quadrada com vários níveis nas colunas
```

```
ks_matrix = ks_df.pivot_table(index='Class 1', columns='Class 2', values=['KS Test Statistic', 'P-Value'], fill_value=0)

# Marque os pares que passaram no teste KS com um asterisco (*) na coluna Estatística do teste KS
# Assumindo 0,05 como nível de significância

passed_test = ks_matrix['P-Value'] >= 0.05
ks_matrix['KS Test Statistic'] = ks_matrix['KS Test Statistic'].astype(str) +
passed_test.applymap(lambda x: '*' if x else '')

print(ks_matrix)
# Converte o ks_matrix em um DataFrame
pd.options.display.float_format = '{:.3f}'.format
ks_matrix_df = pd.DataFrame(ks_matrix)

ks_matrix_df
# Gera o ks_df DataFrame para criar uma matriz quadrada com vários níveis nas colunas
# Arredonde os valores no DataFrame para três casas decimais

ks_matrix = ks_df.pivot_table(index='Class 1', columns='Class 2', values=['KS Test Statistic', 'P-Value'], fill_value=0)
ks_matrix = ks_matrix.round(3)

# Marque os pares que passaram no teste KS com um asterisco (*) na coluna Estatística do teste KS
# Assumindo 0,05 como nível de significância

passed_test = ks_matrix['P-Value'] >= 0.05
ks_matrix['KS Test Statistic'] = ks_matrix['KS Test Statistic'].astype(str) +
passed_test.applymap(lambda x: '*' if x else '')

# Renomeie as colunas para usar "STI" em vez de "Class"
ks_matrix.columns = [f'STI ({col[1]/10})' if col[0] == 'KS Test Statistic' else col[1] for col in
ks_matrix.columns]

# Renomeie o índice para usar "STI" em vez de "Class"
ks_matrix.index = [f'STI ({index/10})' for index in ks_matrix.index]

print(ks_matrix)
# Gere dados de amostra aleatórios para fins ilustrativos
# Carrega o conjunto de dados

df_KPCR = pd.read_csv('/kaggle/input/sti-prediction/dataset_high_z_dim_to_KPCR.csv')
df_KPCR.dropna(inplace=True)

df_KPCR = df_KPCR.groupby(by=["classes"], dropna=True).mean()
df_KPCR = df_KPCR.drop(columns=["Unnamed: 0"])

df_KPCR = df_KPCR.apply(lambda x: x - x.mean())

X_KPCR = df_KPCR.values
y_KPCR = df_KPCR.index.values

X_input = X_KPCR
y_output = y_KPCR/10

# Varie o número de componentes a serem mantidos
n_components_list = [1, 2, 3, 4, 5, 6]

mse_scores = []
```

```
residuals = []

plt.figure(figsize=(16, 12))

for i, n_components in enumerate(n_components_list):
    plt.subplot(3, 2, i + 1)

    kpca = KernelPCA(n_components=n_components, kernel='rbf')
    X_kpca = kpca.fit_transform(X_input)

    regressor = LinearRegression()
    regressor.fit(X_kpca, y_output)

    y_pred = regressor.predict(X_kpca)

    mse = np.mean((y_output - y_pred) ** 2)
    mse_scores.append(mse)

    residual = y_output - y_pred
    residuals.append(residual)

    plt.scatter(y_output, y_pred, label=f'Regressão com componentes={n_components}')
    plt.xlabel('STI esperado')
    plt.ylabel('STI predito')

    m, b = np.polyfit(y_output, y_pred, 1)
    plt.plot(y_output, m * y_output + b, color='red', linestyle='--')
    plt.text(0.1, 0.5, f'y = {m:.2f}x + {b:.2f}', transform=plt.gca().transAxes, fontsize=12, color='red')

    mse_equation = f'MSE = {mse:.2f}'
    plt.legend([f'Regressão com componentes = {n_components}', mse_equation])


plt.tight_layout(pad=1.0)
plt.savefig('Figura_35.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()

# Plota o efeito do número de componentes no MSE separadamente

plt.figure(figsize=(10, 6))

# Plota o efeito do número de componentes no MSE
# Ajusta uma linha de regressão linear aos dados MSE

coefficients = np.polyfit(n_components_list, mse_scores, 1)
mse_fit = np.poly1d(coefficients)

plt.plot(n_components_list, mse_scores, marker='o', label='MSE')
plt.xlabel('Número de componentes')
plt.ylabel('Erro Quadrático Médio (MSE)')

plt.plot(n_components_list, mse_fit(n_components_list), color='red', linestyle='--', label='Linha de
regressão sobre o MSE')

equation = f'MSE = {coefficients[0]:.2f} * componentes + {coefficients[1]:.2f}'
plt.text(0.55, 0.80, equation, transform=plt.gca().transAxes, fontsize=14, color='red')

plt.legend()
plt.tight_layout()
```

```
plt.show()

plt.savefig('Figura_36.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
```

### 10.3 - PERSISTÊNCIA DO MODELO KPCR USADO COMO FUNÇÃO DE PERDA PERSONALIZADA NO ALGORITMO MVPANP

```
# Classe KPCR personalizada

class KPCR:
    def __init__(self, n_components=2, kernel='linear'):
        self.kernel_pca = KernelPCA(n_components=n_components, kernel=kernel)
        self.linear_regression = LinearRegression()

    def fit(self, X, y):
        X_transformed = self.kernel_pca.fit_transform(X)
        self.linear_regression.fit(X_transformed, y)

    def predict(self, X):
        X_transformed = self.kernel_pca.transform(X)
        return self.linear_regression.predict(X_transformed)

X_kpcr, y_kpcr = dataset[['z1', 'z2']].values, dataset['STI'].values

X_train, X_test, y_train, y_test = train_test_split(X_kpcr, y_kpcr, test_size=0.5, random_state=42)

# Crie um pipeline KPCR com StandardScaler

kpcr = Pipeline([
    ('scaler', StandardScaler()),
    ('kpcr', KPCR(n_components=2, kernel='linear'))  # Adjust n_components and kernel as needed
])

kpcr.fit(X_train, y_train)

# Agora você pode usar seu modelo KPCR para previsões e avaliações
# Previsões

y_pred = kpcr.predict(X_test)

# Calcula o (R2)
r2 = r2_score(y_test, y_pred)
print(f"R-squared (R2): {r2}")

# Calcula os valores residuais
residuals = y_test - y_pred

plt.scatter(y_test, y_pred)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], linestyle='--', color='r', label='Linha de
regressão')
plt.xlabel("True Values (STI)")
plt.ylabel("Predictions")
plt.title("True vs. Predicted Values")
plt.legend()
plt.show()
plt.close()
```

```
plt.scatter(y_test, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel("True Values (STI)")
plt.ylabel("Residuals")
plt.title("Residuals")
plt.show()
plt.close()

# Salvando o modelo KPCR
dump(kpcr, 'kpcr_model.joblib')

# Carregando o modelo KPCR
kpcr_persisted = load('kpcr_model.joblib')

# Agora faz a predições com o modelo salvo/persistido

y_pred_persisted = kpcr_persisted.predict(X_test[0:1])
print("Valor previsto usando o modelo KPCR persistente:", y_pred_persisted)
```

## 11 - METODOLOGIA DE TRANSFERÊNCIA DE APRENDIZAGEM

### 11.1 - CARREGAR OS DADOS DO DOMÍNIO DE DESTINO

```
# Função de carregamento do conjunto de dados de teste/validação para aprendizagem por
# transferência
# Carregar dados

A = np.load('../input/sti-prediction/data_to_transfer_02_11_22.npz')
A.files
A = A['arr_0']
df = pd.DataFrame({'c1': A[:,0], 'c2': A[:,1], 'c3': A[:,2], \
                   'c4': A[:,3], 'c5': A[:,4]})

"""## 2 - Configurações de funções
"""

def feature_normalize(dataset):

    mu = np.mean(dataset, axis=0)
    sigma = np.std(dataset, axis=0)
    return (dataset - mu)/sigma


def show_confusion_matrix(validations, predictions):

    matrix = metrics.confusion_matrix(validations, predictions)
    plt.figure(figsize=(6, 4))
    sns.heatmap(matrix,
                cmap="coolwarm",
                linecolor='white',
                linewidths=1,
                xticklabels=LABELS,
                yticklabels=LABELS,
                annot=True,
                fmt="d")

    plt.title("Matriz de confusão")
    plt.ylabel("Classe verdadeira")
    plt.xlabel("Classe predita")
    plt.show()
    plt.close()


def show_basic_dataframe_info(dataframe,
                              preview_rows=20):


    print("Número de linhas do dataframe: %i" % (dataframe.shape[1]))
    print("Número de linhas do dataframe: %i\n" % (dataframe.shape[0]))
    print("Primeiras 20 linhas do dataframe:\n")
    print(dataframe.head(preview_rows))
    print("\nDescrição do dataframe:\n")
```

```
def read_data(file_path):

    """
     Esta função lê os dados de um arquivo
     Argumentos:
         file_path: URL apontando para o arquivo pickle
     Retorna:
         Um dataframe do pandas
    """

    df.rename(columns = {'c1': 'Room-id', 'c2': 'STI',
                         'c3': 'Freq - [Hz]', 'c4': 'PSD(RIR)',
                         'c5': 'PSD(BGN)' },  inplace = True)


    n_bins = 10


    label_encoder = LabelEncoder()
    y = label_encoder.fit_transform(pd.cut(df['STI'], n_bins, retbins=True)[0])

    df['STI'] = y
    df.dropna(axis=0, how='any', inplace=True)

    return df

def convert_to_float(x):

    try:
        return np.float(x)
    except:
        return np.nan


def feature_normalize(dataset):

    mu = np.mean(dataset, axis=0)
    sigma = np.std(dataset, axis=0)
    return (dataset - mu)/sigma


def plot_axis(ax, x, y, title):

    ax.plot(x, y)
    ax.set_title(title)
    ax.xaxis.set_visible(False)
    ax.set_ylim([min(y) - np.std(y), max(y) + np.std(y)])
    ax.set_xlim([min(x), max(x)])
    ax.grid(True)
```

```
def plot_activity(activity, data):

    fig, (ax0, ax1) = plt.subplots(nrows=2,
        figsize=(15, 10),
        sharex=True)
    plot_axis(ax0, data['Freq - [Hz]'], data['PSD(RIR)'], 'PSD(RIR)')
    plot_axis(ax1, data['Freq - [Hz]'], data['PSD(BGN)'], 'PSD(BGN)')
    plt.subplots_adjust(hspace=0.2)
    fig.suptitle(activity)
    plt.subplots_adjust(top=0.90)
    plt.show()
    plt.close()


def create_segments_and_labels(df, freq_steps, step, label_name):

    """
     Esta função recebe um dataframe e retorna os segmentos remodelados
     de dados BGN, RIR, bem como os rótulos STI correspondentes
     Argumentos:
        df: Dataframe no formato esperado
        freq_steps: valor inteiro do comprimento de um segmento criado
     Retorna:
        remodelados segmentos
        rótulos:
     """

    N_FEATURES = 2


    segments = []
    labels = []

    for i in range(0, len(df) - freq_steps, step):
        xs = df['PSD(RIR)'].values[i: i + freq_steps]
        ys = df['PSD(BGN)'].values[i: i + freq_steps]
        xs = xs[1:]
        ys = ys[1:]

        label = stats.mode(df[label_name][i: i + freq_steps])[0][0]
        segments.append([xs, ys])
        labels.append(label)

    reshaped_segments = np.asarray(segments, dtype= np.float32) \
                        .reshape(-1, freq_steps-1, N_FEATURES)
    labels = np.asarray(labels)

    return reshaped_segments, labels


"""
## Criar banco de dados de treinamento
"""

pd.options.display.float_format = '{:.3f}'.format
print('keras version ', keras.__version__)

LABELS = ["1","2","3","4","5","6 ","7","8","9","10"]
```

```
FREQ_PERIODS = len(np.arange(0,8001,10))
STEP_DISTANCE = FREQ_PERIODS

print("\n--- Carregar, inspecionar e transformar dados ---\n")
print("Carregar conjunto de dados do npz")

read_data(df)
show_basic_dataframe_info(df, 20)

print("\n--- Calculando o número total de salas ---\n")
num_rooms = len(df) / FREQ_PERIODS

df['STI'].value_counts().plot(kind='bar', title='')

print("\n--- Remodela os dados em segmentos ---\n")

df_train = df[df['Room-id'] <= 0.7* num_rooms]
df_test = df[df['Room-id'] > 0.7* num_rooms]

scaler = MinMaxScaler()
column_names_to_normalize = ['PSD(RIR)',  'PSD(BGN)']

x = df_train[column_names_to_normalize].values
x_scaled = scaler.fit_transform(x)

df_temp = pd.DataFrame(x_scaled,
                       columns=column_names_to_normalize,
                       index = df_train.index)
df_train[column_names_to_normalize] = df_temp


x = df_test[column_names_to_normalize].values
x_scaled = scaler.fit_transform(x)

df_temp = pd.DataFrame(x_scaled,
                       columns=column_names_to_normalize,
                       index = df_test.index)

df_test[column_names_to_normalize] = df_temp


df_train = df_train.round({'PSD(RIR)': 6, 'PSD(BGN)': 6})
df_test =  df_test.round({'PSD(RIR)': 6, 'PSD(BGN)': 6})


LABEL = "STI"
x_train, y_train = create_segments_and_labels(df_train,
                                              FREQ_PERIODS,
                                              STEP_DISTANCE,
                                              LABEL)

x_test, y_test = create_segments_and_labels(df_test,
                                            FREQ_PERIODS,
                                            STEP_DISTANCE,
                                            LABEL)

plt.close()
```

```
print("\n--- Remodelar dados para serem aceitos pelo módulo Keras ---\n")


print('x_train shape: ', x_train.shape)
print(x_train.shape[0], 'amostras de treinamento')


print('y_train shape: ', y_train.shape)

print('y_test shape: ', y_test.shape)

print("Verificando a dimensão do conjunto de dados de treinamento")

print("Dimensões antes")
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

max_psd_freq = 1600

# len(np.arange(0,8001,10)) aproximação de 800 amostras PDS(RIR) e 800  amostras do PSD(BGN)

print("Dimensão após reestruturação - treino")
x_train_reshaped = x_train.reshape(x_train.shape[0], max_psd_freq)
print(x_train_reshaped.shape)

# Agora para o conjunto de testes
print("Dimensão após reestruturação - teste")
x_test_reshaped = x_test.reshape(x_test.shape[0], max_psd_freq)
print(x_test_reshaped.shape)

x_tr = x_train_reshaped
x_te = x_test_reshaped
```

**11.2 - ESTATÍSTICAS DE DISTRIBUIÇÃO DE DADOS DO DOMÍNIO DE DESTINO**

```
data_target_ANOVA = np.append(y_train, y_test)/10
mu, std = stats.norm.fit(data_target_ANOVA)

plt.hist(data_target_ANOVA, bins=20, alpha=0.6, color='b')

plt.xlabel('STI')
plt.ylabel('Frequência')

print(mu, std)

plt.savefig('Figura_21_b.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()

plt.hist(data_target_ANOVA , bins=20, density=True, alpha=0.6, color='b')

xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, mu, std)

plt.xlabel('STI')
plt.ylabel('Função densidade de probabilidade')

ax2 = plt.twinx()
```

```
ax2.plot(x, p, 'r--')
ax2.set_ylabel('Ajuste distribuição normal', color='r')

plt.show()

print(mu, std)
plt.savefig('Figura_21_d.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
plt.hist(data_target_ANOVA , bins=20, density=True, alpha=0.6, color='b')


xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
c = stats.norm.cdf(x, mu, std)

plt.xlabel('STI')
plt.ylabel('Função densidade de probabilidade')

ax2 = plt.twinx()

ax2.plot(x, c, 'r--')
ax2.set_ylabel('Função densidade acumulada ajustada normal', color='r')

plt.show()

print(mu, std)
plt.savefig('Figura_21_f.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
```

**11.3 - ANOVA DIFERENÇA ENTRE AS MÉDIAS + KS TEST**

```
# Execute o teste KS
ks_statistic, p_value_ks = stats.ks_2samp(data_source_ANOVA, data_target_ANOVA)
alpha = 0.01

# Crae uma figura com duas subtramas
fig, (ax1, ax3) = plt.subplots(1, 2, figsize=(15, 5))

n, bins, patches = ax1.hist(data_source_ANOVA, bins=10, alpha=0.5, label='Origem', color='red',
density=True)
cumulative = np.cumsum(n)
ax3.plot(bins[:-1], cumulative, color='red', linestyle='-', marker='o', label='FDA (Origem)')

source_mean = data_source_ANOVA.mean()
ax1.axvline(source_mean, color='red', linestyle='dashed', linewidth=2, label=f'Média (Origem) =
{source_mean:.2f}')

ax2 = ax1.twinx()

n, bins, patches = ax2.hist(data_target_ANOVA, bins=10, alpha=0.5, label='Alvo', color='blue',
density=True)
cumulative = np.cumsum(n)
ax3.plot(bins[:-1], cumulative, color='blue', linestyle='-', marker='o', label='FDA (Alvo)')

target_mean = data_target_ANOVA.mean()
ax2.axvline(target_mean, color='blue', linestyle='dashed', linewidth=2, label=f'Média (Alvo) =
{target_mean:.2f}')
```

```
# Defina rótulos e legendas para a primeira subtrama
ax1.set_xlabel('STI')
ax1.set_ylabel('Frequência (Origem)')
ax2.set_ylabel('Frequência (Alvo)')

lines1, labels1 = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
lines = lines1 + lines2
labels = labels1 + labels2
ax1.legend(lines, labels, loc='best')

ax3.set_xlabel('STI')
ax3.set_ylabel('Função de Distribuição Acumulada (FDA)')
ax3.legend(loc='best')

plt.grid(False)

# Mostrar resultados do teste KS
if p_value_ks < alpha:
    similarity_result = "Diferença significante"
else:
    similarity_result = "Nenhuma diferença significativa"

plt.show()

print(similarity_result)
print(p_value_ks)
plt.savefig('Figura_22.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
```

**11.4 - VERIFICA SE OS DOMÍNIOS DE ORIGEM E DESTINO SÃO DERIVADOS DE UMA DISTRIBUIÇÃO NORMAL**

```
# Realize o teste Shapiro-Wilk para Fonte
statistic_source, p_value_source = stats.shapiro(data_source_ANOVA)
alpha = 0.01

# Execute o teste Shapiro-Wilk para Target
statistic_target, p_value_target = stats.shapiro(data_target_ANOVA)

# Plotar histogramas e gráficos QQ para ambos os conjuntos de dados
plt.figure(figsize=(12, 10))

# Histograma e gráfico QQ para fonte
plt.subplot(2, 2, 1)
plt.hist(data_source_ANOVA, bins=20, alpha=0.5, label='Origem', color='red')
source_mean = data_source_ANOVA.mean()
plt.axvline(source_mean, color='red', linestyle='dashed', linewidth=2, label=f'Média (Origem) =
{source_mean:.2f}')
plt.legend()
plt.title('Histograma com a média (Origem)')
plt.xlabel('STI')
plt.ylabel('Frequência')
plt.grid(False)

plt.subplot(2, 2, 2)
stats.probplot(data_source_ANOVA, dist="norm", plot=plt)
plt.title('Gráfico Q-Q : (Origem)')
```

```
plt.xlabel('Quantis teóricos')

plt.subplot(2, 2, 3)
plt.hist(data_target_ANOVA, bins=20, alpha=0.5, label='Alvo', color='blue')
target_mean = data_target_ANOVA.mean()
plt.axvline(target_mean, color='blue', linestyle='dashed', linewidth=2, label=f'Média (Alvo) =
{target_mean:.2f}')
plt.legend()
plt.title('Histograma com a média (Alvo)')
plt.xlabel('STI')
plt.ylabel('Frequência')
plt.grid(False)

plt.subplot(2, 2, 4)
stats.probplot(data_target_ANOVA, dist="norm", plot=plt)
plt.xlabel('Quantis teóricos')
plt.title('Gráfico Q-Q : (Alvo)')

plt.tight_layout()

if p_value_source < alpha:
    normality_result_source = " Não normalmente distribuído "
else:
    normality_result_source = " Normalmente distribuído "

if p_value_target < alpha:
    normality_result_target = " Não normalmente distribuído
else:
    normality_result_target = " Normalmente distribuído "


plt.show()

plt.savefig('Figura_34_b.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()


print(f'Resultados do teste Shapiro-Wilk\n\n'
            f'Fonte:\n'
            f'Estatística: {statistic_source:.2f}, valor P: {p_value_source:.2f}\n'
            f'Resultado: {normality_result_source}\n\n'
            f'Alvo:\n'
            f'Estatística: {statistic_target:.2f}, valor P: {p_value_target:.2f}\n'
            f'Resultado: {normality_result_target}')
```

## 12 - RECONSTRUINDO O ESPAÇO LATENTE VAE PARA O DOMÍNIO DE DESTINO

```
# por meio do modelo de codificador treinado no domínio de origem
reconstructed_model = keras.models.load_model("vae_encoder.h5")

# x_tr_latent = encoder.predict(x_tr, batch_size=batch_size)
x_tr_latent = reconstructed_model.predict(x_tr, batch_size=1)
y_tr = y_train
dataset = pd.DataFrame({'z1': x_tr_latent[:, 0], 'z2': x_tr_latent[:, 1], 'STI':  y_tr})
print(dataset)
```

### 12.1 - PLOTANDO A REPRESENTAÇÃO DO ESPAÇO LATENTE (Z = 2) DOS DADOS DE TESTE NO CONJUNTO DE DADOS DE DESTINO

```
# Espaço Latente obtido do domínio alvo
# Plote o domínio alvo (target)

plt.figure(figsize=(6, 6))
y_tr = y_train
plt.scatter(x_tr_latent[:, 0], x_tr_latent[:, 1], c=y_tr/10)
plt.colorbar()

plt.legend(title="STI - Dados de treinamento (Alvo) ")
plt.xlabel('Dimensão - z1')
plt.ylabel('Dimensão - z2')

plt.show()
plt.savefig('Figura_28_b.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
```

#### 12.1.1 - Visualização da Estimativa de Densidade do Kernel (KDE) para o Domínio de Destino

```
for sti_value in range(1, 10):
    ax =  dataset[dataset['STI'] == sti_value][['z1', 'z2']].plot.kde()
    plt.xlabel('Dimensão latente - z')
    plt.ylabel('Densidade de Kernel')
    plt.text(0.03, 0.9, f'STI = {sti_value / 10}', transform=plt.gca().transAxes, fontsize=12, color='black')
    plt.show()
    plt.savefig(f'Figura_29_alvo_{sti_value / 10}.png', dpi=300, bbox_inches='tight', transparent=True)
    plt.close()
```

### 12.1.2 - MAPEAMENTO DOS DADOS DE ALVO (MMD - Target Domain)

```
z_mapping_VAE_target_domain = dataset
z_mapping_VAE_target_domain


def plot_kde_for_sti(sti_value):

    plt.figure(figsize=(6, 6))

    x = dataset[dataset['STI'] == sti_value]['z1']
    y = dataset[dataset['STI'] == sti_value]['z2']

    nbins = 300
    k = kde.gaussian_kde([x, y])
    xi, yi = np.mgrid[x.min():x.max():nbins*1j, y.min():y.max():nbins*1j]
    zi = k(np.vstack([xi.flatten(), yi.flatten()]))

    plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='auto')
    plt.xlabel('Dimensão latente - z1')
    plt.ylabel('Dimensão latente - z2')
    plt.title(f'STI = {sti_value/10}')
    plt.show()
    plt.savefig(f'Figura_32_alvo_{sti_value / 10}.png', dpi=300, bbox_inches='tight', transparent=True)
    plt.close()


for sti_value in [1, 3, 6, 8]:
    plot_kde_for_sti(sti_value)
```

### 12.2 - CÁLCULO DA MÉTRICA MMD E VISUALIZAÇÃO

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

m = 20

x_mean = torch.FloatTensor(([z_mapping_VAE_source_domain["z1"].mean(),
            z_mapping_VAE_source_domain["z2"].mean()]))


y_mean = torch.FloatTensor(([z_mapping_VAE_target_domain["z1"].mean(),
            z_mapping_VAE_target_domain["z2"].mean()]))


x_cov =  torch.FloatTensor(
                    np.cov([z_mapping_VAE_source_domain["z1"],
                        z_mapping_VAE_source_domain["z2"]])
                        ) * torch.eye(2)

# IMPORTANTE: As matrizes de covariância devem ser positivas e definidas
y_cov = torch.FloatTensor(
                    np.cov([z_mapping_VAE_target_domain["z1"],
                        z_mapping_VAE_target_domain["z2"]])
                        ) * torch.eye(2)
```

```
# Fazendo a amostragem
px = MultivariateNormal(x_mean, x_cov)
qy = MultivariateNormal(y_mean, y_cov)

x = px.sample([m]).to(device)
y = qy.sample([m]).to(device)

def MMD(x, y, kernel):

    """Discrepância média máxima empírica. Quanto menor o resultado
       mais evidências de que as distribuições são iguais.

     Argumentos:
        x: primeira amostra, distribuição P
        y: segunda amostra, distribuição Q
        kernel: tipo de kernel como "multiscale" ou "rbf"
     """

    xx, yy, zz = torch.mm(x, x.t()), torch.mm(y, y.t()),torch.mm(x, y.t())

    rx = (xx.diag().unsqueeze(0).expand_as(xx))
    ry = (yy.diag().unsqueeze(0).expand_as(yy))

    dxx = rx.t() + rx - 2. * xx
    dyy = ry.t() + ry - 2. * yy
    dxy = rx.t() + ry - 2. * zz

    XX, YY, XY = (torch.zeros(xx.shape).to(device),
                  torch.zeros(xx.shape).to(device),
                  torch.zeros(xx.shape).to(device))

    if kernel == "multiscale":

        bandwidth_range = [0.2, 0.5, 0.9, 1.3]
        for a in bandwidth_range:
            XX += a**2 * (a**2 + dxx)**-1
            YY += a**2 * (a**2 + dyy)**-1
            XY += a**2 * (a**2 + dxy)**-1

    if kernel == "rbf":

        bandwidth_range = [10, 15, 20, 50]
        for a in bandwidth_range:
            XX += torch.exp(-0.5*dxx/a)
            YY += torch.exp(-0.5*dyy/a)
            XY += torch.exp(-0.5*dxy/a)

    return torch.mean(XX + YY - 2. * XY)


result = MMD(x, y, kernel="multiscale")

print(f"MMD resultado de X e Y é {result.item()}")

# Organização da plotagem
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(12,5), dpi=300)

delta = 0.025
```

```
x1_val = np.linspace(-5, 5, num=m)
x2_val = np.linspace(-5, 5, num=m)

x1, x2 = np.meshgrid(x1_val, x2_val)

px_grid = torch.zeros(m,m)
qy_grid = torch.zeros(m,m)

for i in range(m):
    for j in range(m):
        px_grid[i,j] = multivariate_normal.pdf([x1_val[i],x2_val[j]],
                                               x_mean,
                                               x_cov)

        qy_grid[i,j] = multivariate_normal.pdf([x1_val[i],x2_val[j]],
                                               y_mean,
                                               y_cov)


CS1 = ax1.contourf(x1, x2, px_grid,100, cmap=plt.cm.YlGnBu)
ax1.set_title("Distribuição de $X \sim P(X)$")
ax1.set_ylabel('$x_2$')
ax1.set_xlabel('$x_1$')
ax1.set_aspect('equal')
ax1.scatter(x[:10,0].cpu(), x[:10,1].cpu(),
            label="$X$ ($z_s$ - origem)",
            marker="o",
            facecolor="r",
            edgecolor="k")

ax1.legend()

CS2 = ax2.contourf(x1, x2, qy_grid,100, cmap=plt.cm.YlGnBu)
ax2.set_title("Distribuição de $Y \sim Q(Y)$")
ax2.set_xlabel('$y_1$')
ax2.set_ylabel('$y_2$')
ax2.set_aspect('equal')
ax2.scatter(y[:10,0].cpu(), y[:10,1].cpu(),
            label="$Y$ ($z_t$ - alvo)",
            marker="o",
            facecolor="r",
            edgecolor="k")

ax2.legend()

fig.subplots_adjust(right=0.8)
cbar_ax = fig.add_axes([0.85, 0.15, 0.02, 0.7])
cbar = fig.colorbar(CS2, cax=cbar_ax)
cbar.ax.set_ylabel('Densidade')


plt.show()
plt.savefig('Figura_30.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
```

### 12.3 - PREVISÃO EM LOTE PARA O CONJUNTO DE DADOS X_TR

```
print(x_tr[0].shape)
x_tr_latent = encoder.predict(x_tr, batch_size=batch_size)

# Este resultado será o valor de referência para a função de perda

kpcr.predict(x_tr_latent)

print("Esta foi a previsão em lote")
```

### 12.4 - PREVISÃO DE AMOSTRA ÚNICA NO DOMÍNIO DE DESTINO USANDO O MODELO DE CODIFICADOR TREINADO NO DOMÍNIO DE ORIGEM

```
print(x_tr[0:2].shape)

x_tr_latent = encoder.predict(x_tr[0:2], batch_size=batch_size)
x_tr_latent

kpcr.predict(x_tr_latent)
```

### 12.5 - VALIDAÇÃO DE MODELO: PREDIÇÃO PARA UMA AMOSTRA E VALIDAÇÃO DE DIMENSÃO

```
print("Input sinal bruto - Dimensão bruta antes de formatar:", x_test.shape)

print("Dimensão após reestruturação para servir com a input do autoencoder:",
      x_test.reshape(x_test.shape[0],
      max_psd_freq).shape)

x_test_reshaped = x_test.reshape(x_test.shape[0], max_psd_freq)

# configura a entrada do autoencoder (lote)
x_input_vae = x_test_reshaped

print("Fazendo uma predição no VAE para obter o espaço amostral: ",
    np.expand_dims(x_input_vae[2], axis=0).shape)

x_tr_latent_pred = encoder.predict(np.expand_dims(x_input_vae[5], axis=0),
                                   batch_size=batch_size)

# x_tr_latent = encoder.predict(x_input_vae[2:3], batch_size=batch_size)
x_tr_latent_pred

kpcr.predict(x_tr_latent_pred)
```

## 13 - MODELAGEM DA TRANSFERÊNCIA DE APRENDIZAGEM

```
# verificando a forma
print(x_tr.shape)
print(y_train.shape)
# fix it
X_train, X_test, y_train, y_test = train_test_split(x_tr,
                                                    y_train,
                                                    test_size=0.1,
                                                    random_state=42)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
over_sampler = RandomOverSampler(random_state=42)

X_res, y_res = over_sampler.fit_resample(X_train, y_train)

print(f"Estatísticas de treinamento: {Counter(y_res)}")
X_train, y_train = X_res, y_res/10
X_res_test, y_res_test = over_sampler.fit_resample(X_test, y_test)
print(f"Estatísticas de treinamento: {Counter(y_res_test)}")
X_test, y_test = X_res_test, y_res_test/10
```

## 13.1 - CARGA DA FUNÇÃO DE PERDA KPCR USADA NO ALGORITMO MVPANP

```
# Carregar
kpcr_persisted = load('kpcr_model.joblib')
kpcr.predict(x_tr_latent)

# Carrega o modelo VAE em memória
encoder_reconstruted = keras.models.load_model("codificador_VAE_model.model")

print(x_tr[0:1].shape)
print(type(x_tr[0:1]))
print(x_tr[0:1].dtype)

latent_pred = encoder_reconstruted.predict(x_tr[0:2], batch_size=1)
latent_pred
kpcr.predict([[1., 1.0]])
```

### 13.2 - IMPLEMENTAÇÃO DA FUNÇÃO KPCR + MVPANP COMO PERDA NO MODO EAGER

```
kl = tf.keras.losses.KLDivergence()

# Reconstruindo o modelo VAE treinado no domínio de origem

encoder_reconstruted = keras.models.load_model("codificador_VAE_model.model")

def KPCR_Prediction_Loss(input_tensor):

    z_latent = tensorflow_graph(np.expand_dims(input_tensor, axis=0),
                                False, None).numpy()

    src_pred = kpcr.predict(z_latent)

    return src_pred


@tf.function(input_signature=[tf.TensorSpec(None, tf.float32)])

def KPCR_loss(input):

    y = tf.numpy_function(KPCR_Prediction_Loss, [input], tf.float32).numpy()

    return y


def MVPAnP_Loss(data, y_pred):

    y_true = data[0][0]
    input2latent = data[0][1:1601]

    return K.mean(K.square(y_pred - y_true),
                  axis=-1) - 0.1* kl(KPCR_loss(input2latent),
                                     y_true)
```

### 13.3 - MODELO MLP (MULTILAYER PERCEPTRON) PARA TESTAR A FUNÇÃO DE PERDA PERSONALIZADA

```
tf.config.run_functions_eagerly(True)

i = Input(shape=(1600,))
x = Dense(50, kernel_initializer='glorot_uniform', activation='relu')(i)
x = Dense(50, activation='relu')(x)
x = Dense(25, activation='sigmoid')(x)
o = Dense(1, activation='sigmoid')(x)
model = Model(i, o)

model.compile(loss=MVPAnP_Loss, optimizer='adam')

history = model.fit(X_train, np.column_stack((y_train, X_train)),
                    epochs=5,
                    batch_size=1,
                    verbose=0)
type(X_train)
print(X_train.shape)
print(y_train.shape)
```

```
data = np.column_stack((y_train, X_train))
data.shape
print(history.history.keys())

plt.plot(history.history['loss'])
plt.title('Função de perda')
plt.ylabel('Perda')
plt.xlabel('Época')
plt.legend(['treino', 'teste'], loc='upper left')
plt.show()
plt.close()

# Avalia a qualidade do modelo
score = model.evaluate(X_train, np.column_stack((y_train, X_train)), verbose=1)
print('Test loss:', score)

# Exemplo de uma predição
yhat = model.predict(X_train)
yhat.shape

print(y_train)
print(y_test)
```

## 13.4 - VALIDAÇÃO DO ALGORITMO MVPANP: USANDO OTIMIZAÇÃO DE HIPERPARÂMETROS E ANOVA

### 13.4.1 - Otimização de Hiperparâmetros para Modelo de Base (1D CONVNET) - SEM Transfer Learning

```
def start_Baseline(features1,features2, trials,  plot_graph = False):

 # MODELO 1DCONVNET
 def create_model(activation,
                  dropout_rate,
                  filters,
                  kernel_size,
                  strides,
                  kernel_initializer):


   """
    Cria um modelo temporário em memória
   """

   clear_session()


   if type(filters) == tuple:
      filter_num = filters[0]
   else:
      filter_num = filters


   i = Input(shape=(1600,1))
   x = Conv1D(filters=filter_num,
              kernel_size=kernel_size,
              strides=strides,
              kernel_initializer=kernel_initializer,
              activation=activation,
              input_shape=(1600, 1))(i)
   x = Conv1D(100, 10, activation='relu')(x)
   x = Conv1D(100, 10, activation='relu')(x)
   x = MaxPooling1D(3)(x)
   x = Conv1D(160, 10, activation='relu')(x)
   x = Conv1D(160, 10, activation='relu')(x)
   x = GlobalAveragePooling1D()(x)
   x = Dropout(rate=dropout_rate)(x)
   x = Dense(40, activation='relu')(x)
   o = Dense(1, activation='sigmoid')(x)

   ### - 1D CONV NET MODEL finaliza

   model = Model(i, o)
   model.compile(loss='mean_squared_error', optimizer='adam')

   return model

 # Função objetivo para otimizar por OPTUNA
```

```
def objective(trial):

  activation = trial.suggest_categorical("activation",
                                         ["relu", "sigmoid", "swish"]
                                        )
  dropout_rate = trial.suggest_float("dropout", 0.1, 0.3)
  num_filters = trial.suggest_categorical("filters", [32, 64]),
  kernel_size = trial.suggest_categorical("kernel_size", [3, 5]),
  strides = trial.suggest_categorical("strides", [1, 2]),

  if (activation == "relu"):

     model = create_model(activation,
                          dropout_rate=dropout_rate,
                          filters=num_filters,
                          kernel_size=kernel_size,
                          strides=strides,
                          kernel_initializer="HeUniform")
  else:
     model = create_model(activation,
                          dropout_rate=dropout_rate,
                          filters=num_filters,
                          kernel_size=kernel_size,
                          strides=strides,
                          kernel_initializer="GlorotUniform",)

  # Implementar critério de parada antecipada.
  # O processo de treinamento para quando há
  # nenhuma melhoria durante 50 iterações

  callback = keras.callbacks.EarlyStopping(monitor='loss',
                                           patience=50
                                          )
  history = model.fit(features1,
                      features2,
                      batch_size = 1,
                      epochs=3,
                      callbacks = [callback],
                      verbose = 0)

  return history.history["loss"][-1]

study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=trials)

# Crie o modelo final com os melhores hiperparâmetros
print('Melhores hiperparâmetros encontrados via Optuna: \n', study.best_params)
if (study.best_params['activation'] == "relu"):

  model = create_model(study.best_params['activation'],
                       study.best_params['dropout'],
                       study.best_params['filters'],
                       study.best_params['kernel_size'],
                       study.best_params['strides'],
                       kernel_initializer="HeUniform",
                      )
else:
  model = create_model(study.best_params['activation'],
```

```
                    study.best_params['dropout'],
                    study.best_params['filters'],
                    study.best_params['kernel_size'],
                    study.best_params['strides'],
                    kernel_initializer="GlorotUniform",)

  model.summary()

  # Implementa o critério de parada antecipada.
  # O processo de treinamento para quando não há melhoria durante 50 iterações

  callback = keras.callbacks.EarlyStopping(monitor='loss', patience=50)
  history = model.fit(features1,
                      features2,
                      batch_size = 1,
                      epochs=3,
                      callbacks = [callback],
                      verbose = 0)

  result = model.predict(features1)


  print(f'RMSE Autoencoder: {np.sqrt(mean_squared_error(features2, result))}')
  print('')

  # Os seguintes valores são retornados: extract_f || MSE || Melhores hiperparâmetros OPTUNA

  return mean_squared_error(features2, result), study.best_params, study

AutoEncoder_MSE, AutoEncoder_hyperparams, Study = start_Baseline(X_train, y_train,
                            trials = 10,
                            plot_graph=True)
#  salva resultados
df_results_without_TF = Study.trials_dataframe()
df_results_without_TF

# df_results.to_pickle(results_directory + 'df_optuna_results.pkl')
# df_results.to_csv(results_directory + 'df_optuna_results.csv')
print("Número de testes concluídos: {}".format(len(Study.trials)))

print(" Configuração ótima:")
trial = Study.best_trial

print("  Valor: {}".format(trial.value))

print("  ParamEtros: ")
for key, value in trial.params.items():
    print("    {}: {}".format(key, value))
```

### 13.4.2 - Otimização de hiperparâmetros para modelo base (1D CONVNET) - COM Transfer Learning

```
def start_Benchmarking(features1,features2, trials,  plot_graph = False):

  # Modelo 1DCONVNET
  def create_model(activation,dropout_rate,
                   filters, kernel_size,
                   strides, kernel_initializer):

    clear_session()

    kl = tf.keras.losses.KLDivergence()
    encoder_reconstruted = keras.models.load_model("codificador_VAE_model.model")

    def KPCR_Prediction_Loss(input_tensor):
        z_latent = tensorflow_graph(np.expand_dims(input_tensor, axis=0),
                                    False, None).numpy()
        src_pred = kpcr.predict(z_latent)

        return src_pred

    @tf.function(input_signature=[tf.TensorSpec(None, tf.float32)])
    def KPCR_loss(input):
        y = tf.numpy_function(KPCR_Prediction_Loss, [input], tf.float32).numpy()
        return y

    def MVPAnP_Loss(data, y_pred):
        y_true = data[0][0]
        input2latent = data[0][1:1601]
        return K.mean(K.square(y_pred - y_true), axis=-1) - 0.1*kl(KPCR_loss(input2latent), y_true)

    tf.config.run_functions_eagerly(True)

    # Bug fix
    if type(filters) == tuple:
        filter_num = filters[0]
    else:
        filter_num = filters

    i = Input(shape=(1600,1))
    x = Conv1D(filters=filter_num, kernel_size=kernel_size, strides=strides,
               kernel_initializer=kernel_initializer,
               activation=activation, input_shape=(1600, 1))(i)
    x = Conv1D(100, 10, activation='relu')(x)
    x = Conv1D(100, 10, activation='relu')(x)
    x = MaxPooling1D(3)(x)
    x = Conv1D(160, 10, activation='relu')(x)
    x = Conv1D(160, 10, activation='relu')(x)
    x = GlobalAveragePooling1D()(x)
    x = Dropout(rate=dropout_rate)(x)
    x = Dense(40, activation='relu')(x)
    o = Dense(1, activation='sigmoid')(x)
    model = Model(i, o)

    model.compile(loss=MVPAnP_Loss, optimizer='adam')

    return model
```

```
# Função objetivo para otimizar por OPTUNA
def objective(trial):
  activation = trial.suggest_categorical("activation", ["relu", "sigmoid", "swish"])
  dropout_rate = trial.suggest_float("dropout", 0.1, 0.3)
  num_filters = trial.suggest_categorical("filters", [32, 64]),
  kernel_size = trial.suggest_categorical("kernel_size", [3, 5]),
  strides = trial.suggest_categorical("strides", [1, 2])

  if (activation == "relu"):
    model = create_model(activation,
                         dropout_rate=dropout_rate,
                         filters=num_filters,
                         kernel_size=kernel_size,
                         strides=strides,
                         kernel_initializer="HeUniform")
  else:
    model = create_model(activation,
                         dropout_rate=dropout_rate,
                         filters=num_filters,
                         kernel_size=kernel_size,
                         strides=strides,
                         kernel_initializer="GlorotUniform")

  # Implementar critério de parada antecipada.
  # O processo de treinamento para quando não há melhoria durante 50 iterações

  callback = keras.callbacks.EarlyStopping(monitor='loss', patience=50)
  history = model.fit(features1,
                      features2,
                      batch_size = 1,
                      epochs=3,
                      callbacks = [callback],
                      verbose = 0)

  return history.history["loss"][-1]

study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=trials)

# Crie o modelo final com os melhores hiperparâmetros

print('Melhores hiperparâmetros encontrados via Optuna: \n', study.best_params)
if (study.best_params['activation'] == "relu"):
  model = create_model(study.best_params['activation'],
                       study.best_params['dropout'],
                       study.best_params['filters'],
                       study.best_params['kernel_size'],
                       study.best_params['strides'],
                       kernel_initializer="HeUniform",
                      )
else:
  model = create_model(study.best_params['activation'],
                       study.best_params['dropout'],
                      study.best_params['filters'],
                       study.best_params['kernel_size'],
                       study.best_params['strides'],
                       kernel_initializer="GlorotUniform")

model.summary()
```

```
  # Implementa o critério de parada antecipada.
  # O processo de treinamento para quando não há melhoria durante 50 iterações

  callback = keras.callbacks.EarlyStopping(monitor='loss', patience=50)
  history = model.fit(features1,
                      features2,
                      batch_size = 1,
                      epochs=3,
                      callbacks = [callback],
                      verbose = 0)

  result = model.predict(features1)

  # Resultados
  print(f'RMSE Autoencoder: {np.sqrt(mean_squared_error(features2[:,0], result))}')

  # Os seguintes valores são retornados: extract_f || MSE || OPTUNA melhor hiperparâmetros
reprodutibilidade

  return mean_squared_error(features2[:,0], result), study.best_params, study

AutoEncoder_MSE, AutoEncoder_hyperparams, Study = start_Benchmarking(X_train,
                                                  np.column_stack((y_train,
                                                                   X_train)),
                                trials = 10,
                                plot_graph=True)
df_results_with_TF = Study.trials_dataframe()

# Salvando para reprodutibilidade
# df_results.to_pickle(results_directory + 'df_optuna_results.pkl')
# df_results.to_csv(results_directory + 'df_optuna_results.csv')

pd.options.display.float_format = '{:.5f}'.format

df_results_with_TF
df_results_with_TF["value"].values
print("Número de testes concluídos: {}".format(len(Study.trials)))

print("Rodada ótima :")
trial = Study.best_trial

print("  Valor: {}".format(trial.value))

print("  Parâmetros: ")
for key, value in trial.params.items():
    print("    {}: {}".format(key, value))
# cria um modelo preditivo como exemplo

tf.config.run_functions_eagerly(True)

i = Input(shape=(1600,))
x = Dense(50, kernel_initializer='glorot_uniform', activation='relu')(i)
x = Dense(50, activation='sigmoid')(x)
x = Dense(25, activation='sigmoid')(x)
o = Dense(1, activation='sigmoid')(x)
model = Model(i, o)

model.compile(loss=MVPAnP_Loss, optimizer='adam')
```

```
history = model.fit(X_train, np.column_stack((y_train, X_train)),
                    epochs=50,
                    batch_size=5,
                    verbose=1)
 y_test
y_pred = model.predict(X_test)
mspe = np.mean(((y_test - y_pred) / (y_test + 1)) ** 2) * 100

plt.figure(figsize=(10, 5))
plt.scatter(y_test, y_pred, c='blue', label='STI predito vs. esperados')
plt.xlabel("STI valores de teste - ver seção 3.7.2")
plt.ylabel("STI valores estimados")
plt.xlim([0, 0.8])
plt.ylim([0, 0.8])

regression_model = LinearRegression()
regression_model.fit(y_test.reshape(-1, 1), y_pred)
plt.plot(y_test, regression_model.predict(y_test.reshape(-1, 1)), color='red', linewidth=2,
label='Regressão linear')

equation = "MSPE = {:.2f}%".format(mspe)
plt.text(0.3, 0.8, equation, transform=plt.gca().transAxes, fontsize=15, color='red')
plt.legend()
plt.grid(False)
plt.show()
plt.savefig('Figura_39_A.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()


percentage_error = (((y_pred.flatten() - y_test) / (y_test + 0.1)) ** 2) * 100


plt.figure(figsize=(10, 5))
plt.hist(percentage_error, bins=20, edgecolor='k', label='Histograma de MSPE')
plt.xlabel("Erro percentual relativo (%)")
plt.ylabel("Frequência")
plt.grid(False)
plt.legend()
plt.show()
plt.close()
y_pred = model.predict(X_test)

rmspe = np.sqrt(np.mean(((y_test - y_pred) / (y_test + 1)) ** 2)) * 100

plt.figure(figsize=(10, 5))
plt.scatter(y_test, y_pred, c='blue', label='STI predito vs. esperados')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.xlabel("STI valores de teste - ver seção 3.7.2")
plt.ylabel("STI valores estimados")

regression_model = LinearRegression()
regression_model.fit(y_test.reshape(-1, 1), y_pred)

equation = "RMSPE = {:.2f}%".format(rmspe)
plt.text(0.3, 0.8, equation, transform=plt.gca().transAxes, fontsize=15, color='red')

plt.plot(y_test, regression_model.predict(y_test.reshape(-1, 1)), color='red', linewidth=2,
```

```
label='Regressão Linear')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.legend()
plt.grid(False)
plt.show()

plt.savefig('Figura_39_B.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()

percentage_error = (((y_pred.flatten() - y_test) / (y_test + 0.1)) ** 2) * 100

plt.figure(figsize=(10, 5))
plt.hist(percentage_error, bins=20, edgecolor='k', label='Histograma de RMSPE')
plt.legend()
plt.xlabel("Erro percentual relativo (%)")
plt.ylabel("Frequência")
plt.grid(False)
plt.show()
plt.close()
```

**13.5 - ANÁLISE ANOVA PARA O MODELO DE LINHA DE BASE ANTES E DEPOIS DO TF**

```
df_results_with_TF["Transfer_Learning"] = "Com TF"

df_results_without_TF["Transfer_Learning"] = "Sem TF"
# Faz o merge dos dados com e sem TF
benchmarking_analysis = pd.concat([df_results_without_TF, df_results_with_TF])
benchmarking_analysis['value'] = benchmarking_analysis['value'].values.round(4)
benchmarking_analysis['value'].values.round(4)
```

## 14 - COMPARAÇÃO DOS RESULTADOS DOS MÉTODOS PROPOSTOS DE TF

```
benchmarking_analysis
xx_com_TF = benchmarking_analysis[benchmarking_analysis['Transfer_Learning'] == 'Sem TF']['value'].values
xy_sem_TF = benchmarking_analysis[benchmarking_analysis['Transfer_Learning'] == 'Com TF']['value'].values

# Implementa a ANOVA
f_oneway(xx_com_TF, xy_sem_TF)
```

### 14.1 - ANÁLISE ANOVA E BOXPLOT

```
# Configurando os dados
data = [xx_com_TF,xy_sem_TF]
plt.rcParams['font.size'] = 14
plt.rcParams['axes.linewidth'] = 1.5
plt.rcParams['xtick.major.width'] = 1.5
plt.rcParams['ytick.major.width'] = 1.5

fig, ax1 = plt.subplots(figsize=(10, 5))

bp = ax1.boxplot([xx_com_TF, xy_sem_TF], patch_artist=True)

colors = ['lightblue', 'lightgreen']
for patch, color in zip(bp['boxes'], colors):
    patch.set_facecolor(color)

ax1.set_xticklabels(['Sem Transferência', 'Com Transferência'])
ax1.grid(axis='y', linestyle='--', alpha=0.7)
ax1.set_ylabel('Erro quadrático médio')

ax2 = ax1.twinx()

mu_xx, sigma_xx = np.mean(xx_com_TF), np.std(xx_com_TF)
x_xx = np.linspace(mu_xx - 3 * sigma_xx, mu_xx + 3 * sigma_xx, 100)
pdf_xx = norm.pdf(x_xx, mu_xx, sigma_xx)
cdf_xx = norm.cdf(x_xx, mu_xx, sigma_xx)
ax2.plot(x_xx, pdf_xx, color='red', linestyle='--', label='Dist. Normal (Sem TF)')
ax2.plot(x_xx, cdf_xx, color='blue', linestyle=':', label='FDA (Sem TF)')

mu_xy, sigma_xy = np.mean(xy_sem_TF), np.std(xy_sem_TF)
x_xy = np.linspace(mu_xy - 3 * sigma_xy, mu_xy + 3 * sigma_xy, 100)

pdf_xy = norm.pdf(x_xy, mu_xy, sigma_xy)
cdf_xy = norm.cdf(x_xy, mu_xy, sigma_xy)

ax2.plot(x_xy, pdf_xy, color='purple', linestyle='--', label='Dist. Normal (Com TF)')
ax2.plot(x_xy, cdf_xy, color='orange', linestyle=':', label='FDA (Com TF)')
ax2.legend(loc='upper right')
ax2.set_ylabel('Probabilidade / FDP / FDA', color='red')
ax2.tick_params(axis='y', labelcolor='red')

plt.show()
plt.savefig('Figura_37.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
```

```
plt.rcParams['font.size'] = 14
plt.rcParams['axes.linewidth'] = 1.5
plt.rcParams['xtick.major.width'] = 1.5
plt.rcParams['ytick.major.width'] = 1.5

fig, ax = plt.subplots(figsize=(10, 5))

ax.hist(xx_com_TF, bins=100, cumulative=True, density=True, histtype='step', color='blue', label='FDA (Sem TF)')
ax.hist(xy_sem_TF, bins=100, cumulative=True, density=True, histtype='step', color='orange', label='FDA (Com TF)')
ax.legend(loc='lower right')

f_statistic, p_value = f_oneway(xx_com_TF, xy_sem_TF)
ks_statistic, ks_p_value = ks_2samp(xx_com_TF, xy_sem_TF)

result_table = pd.DataFrame({
    'Test': ['ANOVA', 'KS Test'],
    'p-value': [p_value, ks_p_value],
    'Significance Level': [0.05, 0.05]
})


ax.set_xlabel('Erro quadrático médio - EQM/MSE')
ax.set_ylabel('Função distribuição acumulada - FDA')

plt.show()

print("Resultados de testes estatísticos:")
print(result_table)

plt.savefig('Figura_38.png', dpi=300, bbox_inches='tight', transparent=True)
plt.close()
tukey = pairwise_tukeyhsd(endog=benchmarking_analysis['value'], groups=benchmarking_analysis['Transfer_Learning'], alpha=0.05)
print(tukey)
```

**14.2 - SALVA TODOS OS ARQUIVOS DE GRÁFICOS E MATRIZES DE PESOS**

```
!zip -r graficos_tese_pesos_2.zip /kaggle/working/
```