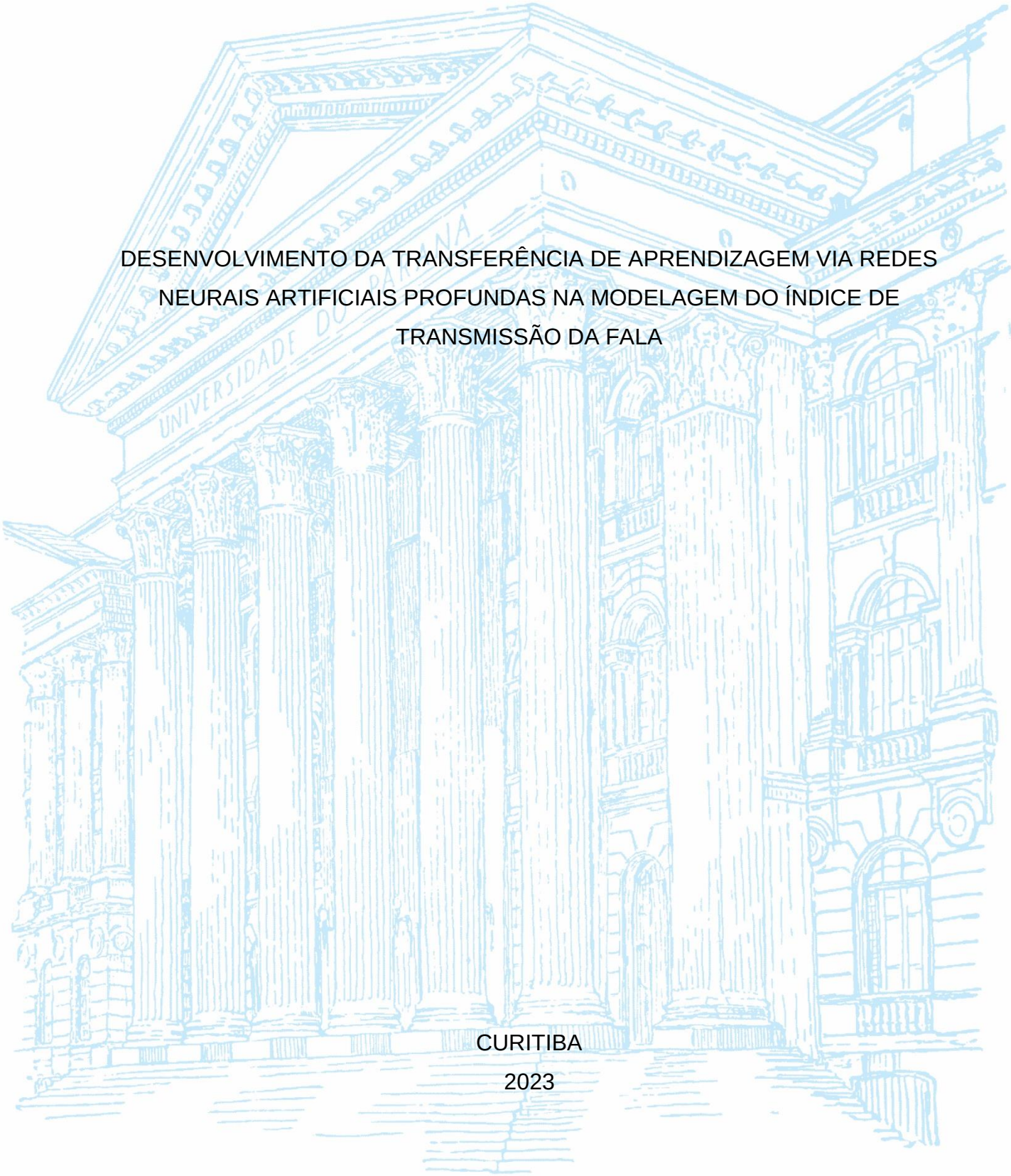
UNIVERSIDADE FEDERAL DO PARANÁ

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 curso 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

TERMO DE APROVAÇÃO

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 curso de Pós-Graduação em Engenharia Mecânica, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título Doutor em Engenharia Mecânica.

_____

Prof. Tit. Dr. -Ing. Paulo Henrique Trombetta Zannin
Orientador – Departamento de Engenharia Mecânica, UFPR

_____

Prof. Dr. Nilson Barbieri
Departamento de Escola Politécnica, PUCPR

_____

Prof. Ph.D. Arcanjo Lenzi
Departamento de Engenharia Mecânica, UFSC

_____

Prof. Ph.D. Eduardo Márcio de Oliveira Lopes
Departamento de Engenharia Mecânica, UFPR

Curitiba, 16 de junho de 2023

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 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, lisura serão sempre guardados comigo.

Ao Prof. Lucas Nonato de Oliveira, e a Profa. Linda Viola Ehlin Caldas que desde a iniciação científica estivarem 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, de 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 para ao secretário do programa, Jonatas Ricardo Zanoto, por todo o auxílio, e ótimas conversas.

A bolsa de estudos do PGMEC, fomentada pela coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES.

A Universidade Federal do Paraná (UFPR) pela ótima estrutura, e o Prof. Zannin pela ótima estrutura do Laboratório de Acústica Ambiental, Industrial e Conforto Acústico – LAAICA, o qual 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 é um elemento fundamental que interfere na dinâmica do ensino-aprendizagem. Dessa forma, salas com acústica deficitária podem onerar socioeconomicamente toda uma sociedade, uma vez que, a função pedagógica perde sua eficiência. A inteligibilidade da fala é mensurada através do Índice de Transmissão da Fala, ou em inglês, *Speech Transmission Index* (STI). Contudo, a medição desse é complexa e requer instrumentação de alto custo. Na literatura corrente há modelos preditivos do STI que usam o Tempo de Reverberação como variável de regressão. Entretanto, esses modelos não ponderam os efeitos espectrais da localização dual tempo-frequência do sinal da resposta impulsiva das salas, assim como o conteúdo espectral do ruído de fundo. Consequentemente, esse trabalho objetivou modelar o STI ao aplicar a rede neural convolucional-unidimensional profunda. A rede neural possuiu duas entradas, a resposta impulsiva simulada do TR via o Método das Fontes Virtuais com 10 000 amostras, a segunda entrada foi o ruído de fundo com 564 amostras. O alvo de treinamento foi o STI preditivo fornecido pela normativa IEC 60268-16. Em face ao entrave da aquisição das medições *in situ*, e como o objetivo de prover robustez as estimativas da rede neural, propôs-se um novo método de Transferência de Aprendizagem denominado de transferência de aprendizagem via Minimização Variacional Projetiva-Adaptativa não paramétrica (MVPAnP). O MVPAnP tem como objetivo otimizar os modelos de rede neural previamente treinado com dados simulados e transferir a generalização para um conjunto de dados de medições *in situ*. A validação dos modelos foi realizada como base na aquisição experimental de 60 salas com medições somente do TR, e com 15 salas com medições do STI e TR. Após, avaliou-se a qualidade do modelo via curva de calibração para a validar experimentalmente o modelo. Os resultados esperados consistem no desenvolvimento de uma nova modelagem para predição do STI em relação aos modelos atuais, tendo em vista os custos e da consequente inviabilidade orçamentária para aquisição da instrumentação da medição do STI. Dessa forma, conclui-se que, o modelo proposto poderá ser utilizado como uma ferramenta de suporte para uma avaliação diagnóstica do STI em salas de aula, usando apenas a instrumentação que é utilizada para as medições do TR.

Palavras-chave: Inteligibilidade da Fala. Índice de Transmissão da Fala. Redes Neurais Convolucionais Profundas. Transferência de Aprendizagem. Acústica de salas.

ABSTRACT

Acoustic comfort in classrooms is a fundamental element that interferes in the teaching-learning dynamics. In this way, rooms with poor acoustics can socioeconomically burden an entire society, since the pedagogical function loses its efficiency. Speech intelligibility is measured using the Speech Transmission Index (STI). However, its measurement is complex and requires expensive instrumentation. In the current literature, there are STI predictive models that use the Reverberation Time as a regression variable. However, these models do not consider the spectral effects of the dual time-frequency location of the impulsive response signal of the rooms, as well as the spectral content of the background noise. Consequently, this work aimed to model the STI by applying the deep one-dimensional convolutional neural network. The neural network had two inputs, the simulated TR impulse response via the Virtual Source Method with 10,000 samples, the second input was the background noise with 564 samples. The training target was the predictive STI provided by the IEC 60268-16 standard. In view of the obstacle of acquiring measurements in situ, and with the objective of providing robustness to the neural network estimates, a new method of Transfer of Learning called transfer of learning via non-parametric Variational Projective-Adaptive Minimization (MVPAnP) was proposed. MVPAnP aims to optimize neural network models previously trained with simulated data and transfer the generalization to a dataset of in situ measurements. Model validation was performed based on the experimental acquisition of 60 rooms with TR measurements only, and with 15 rooms with STI and TR measurements. Afterwards, the quality of the model was evaluated via a calibration curve to experimentally validate the model. The expected results consist of the development of a new model for predicting the STI in relation to current models, considering the costs and the consequent budget impracticability for the acquisition of instrumentation for measuring the STI. Thus, it is concluded that the proposed model can be used as a support tool for a diagnostic evaluation of STI in classrooms, using only the instrumentation that is used for RT measurements.

Keywords: Speech Transmission Index. Artificial Neural Networks. Design of Experiments. Room Acoustics. Reverberation Time.

LISTA DE FIGURAS

LISTA DE QUADROS E TABELAS

LISTA DE ABREVIATURAS OU SIGLAS

TR - Tempo de Reverberação
AFE - Automatic Feature Extraction
PAIR - Perda Auditiva Induzida por Ruído
STI - Speech Transmission Index
ABNT - Associação Brasileira de Normas Técnicas
ANSI - American National Standard Institute
IEC - International Electrotechnical Commission
ISO - International Organization for Standardization
MLP - Multilayer Perceptron
MTF - Modulation Transfer Function
PCA - Principal Component Analysis
RF - Ruído de Fundo
RSR - Relação Sinal Ruído
SNR - Signal to Noise Ratio
SLITs - Sistemas lineares e invariantes no tempo
JND - Just Noticeable Difference
ESTI - Índice de transmissão de Fala Estendido
ISM - Image Source Method
wav - WAVEform audio format
AFE - Extração Automática de Características
Conv1D - Camada de Rede Neural de Convolução unidimensional
DMP - Diferença Minimamente Perceptível
MMD - Maximum Mean Discrepancy
KPCA - Kernel Principal Component Analysis
ACP - Análise de Componentes Principais
VAE - Variational autoencoders
MVPAnP - Minimização Variacional Projetiva-Adaptativa não paramétrica
KPCR - kernel Principal Component Regression
PCR - Principal Component Regression
MAPE - Mean Absolute Percentage Error

LISTA DE SÍMBOLOS

© - Copyright

® - Marca registrada

AL_{cons} - *Articulation Loss of Consonants*

Σ - Somatório de números

L_P - Nível de Pressão Sonora (dB)

p_{ref} - Pressão de referência

$L_{eq,T}$ - Nível de pressão sonora contínuo equivalente (dB)

L_{eq} - Nível de pressão sonora equivalente

$E(t)$ - Energia da curva de decaimento

C50 - Claridade (dB)

D50 - Definição

LF - Distribuição Espacial Lateral

G - Intensidade/Força (dB)

LAeq - Nível de Pressão Sonora Equivalente ponderado em A

U50 - Razão de Som Útil-a-Prejudicial

SUMÁRIO

1 INTRODUÇÃO

O ruído como estressor ambiental apresenta-se como uns 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. Este interfere na relação ensino aprendizagem e pode causar nos alunos e professores doenças correlatas a prolongada exposição (BASNER et al., 2014).

Em ambientes escolares, Connolly (2019) esclareceu os principais efeitos nocivos do ruído no desenvolvimento psicossocial das crianças, neste estudo ainda foi elencado que tais efeitos podem pendurar até as fases posteriores do desenvolvimento da criança, chegando a emergir manifestações na fase adulta. No contexto do sistema auditivo existe relatos da manifestação de Zumbido (*tinnitus*), em casos mais severos, há relatos de Perda Auditiva Induzida por Ruído (Pair). Enquanto, 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, condições acústicas deficitárias tendem a contribuir negativamente como 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 Van Schoonhoven, Rhebergen e Dreschler (2017) o tempo de reverberação por si só apresenta várias desvantagens quando se trata em

definir uma interpretação quanto a qualidade da fala, que é representada pelos descritores acústicos da inteligibilidade da fala. Uma vez 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, rápidos e práticos são de suma importância. Portanto, este trabalho objetiva contribuir com o estado na arte no 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 as 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, e algumas delas 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, *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.

1.2 HIPÓTESE

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

Os modelos correntes utilizam o tempo de reverberação em bandas de oitava como variável de regressão. De tal modo, a modelagem via redes neurais artificiais de aprendizagem profunda convolucionais, propiciam o desenvolvimento de um modelo, que usa todo o sinal captado da resposta impulsiva da sala, advindo da medição direta do TR. Com isso, ao otimizar as redes neurais com a heurística de transferência de aprendizagem, esse trabalho vem 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 objetivará desenvolver um novo método aplicando Redes Neurais Artificiais Profundas acoplado com a heurísticas de transmissão de aprendizagem, por meio da otimização da similaridade de subespaços latentes via projeções com o intuito de prever o STI.

1.3.2 Objetivos específicos

Para atingir o objetivo geral foram elencados os objetivos específicos que foram realizados na seguinte ordem sistematizada, com a seguir:

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 Fontes Virtuais 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 aplicado nas respostas impulsivas e no ruído de fundo;
g) desenvolver o método de transferência de aprendizagem via projeções em espaço 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 nesse 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. No capítulo 1 apresenta a introdução do trabalho. Nesse foram mostradas a justificativas e hipóteses norteadoras para a proposto de desenvolvimento da tese.

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

No capítulo 3 foi desenvolvida a metodologia do trabalho, onde definiu-se os modelos de rede neural convolucional unidimensional. Apresentou-se o algoritmo de treinamento e a respetiva hiper parametrização de treinamento. Também a forma de criação e aquisição do conjunto de dados de treinamento e validação. Em seguida, foi descrito método de aprendizagem por transferência.

O capítulo 4 apresenta os resultados e discussões. O capítulo 5 consolida as considerações finais e delineou-se as sugestões para trabalhos futuros. Anexo apresentação o compilado de todos os códigos desenvolvimentos, para a implementação do método proposto.

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 e tal ambiente exigem modelos que devam incluir os fatores biológicos do som, como a 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 a 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, pode-se retirar abstrações que permitem avaliar a acústica de salas sob a perspectiva quantitativa, representada pela teoria de sistemas lineares invariantes no tempo.

A equação da onda é uma equação diferencial parcial do tipo parabólica de segunda ordem linear (EVANS, 1997). Fisicamente, Vorländer e Summers (2007) realizaram as ponderações energéticas da propagação do som, ao acoplar as 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,

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

em que p é a pressão, c é a velocidade do som no meio, ρ é a massa específica do meio de transmissão, q é a quantidade de movimento, x e o vetor posição e t é o tempo. A Eq. (1) pode ser generalizada para o espaço n-dimensional, dado como,

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

em que Δ é o operador Laplaciano, $\ddot{p}$ representa a segunda derivada temporal da pressão, $\dot{q}$ representa a primeira derivada temporal da quantidade de movimento. O operador laplaciano 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. Notadamente, os candidatos a solução de Eq. (1) dependem das imposições das condições iniciais e de contorno do problema, ponderando ainda se necessário simplificações de modelo.

Para a formulação da solução da equação da onda, adotou-se a abordagem via mapeamento funcional no domínio da frequência (VORLÄNDER; SUMMERS, 2007), nesse caso a solução da Eq. (1) é dado por,

$$p_\omega(r) = iQc^2\omega\rho \sum_n \frac{p_n(r)p_n(r_0)}{(\omega^2 - \omega_n^2 - 2i\delta_n\omega_n)k_n} \tag{4}$$

em que, k_n é o fator que se relaciona com a impedância acústica no meio, considerando o efeito da frequência angular, ω. Estes valores correspondem aos autovalores dos modos de vibrar. O fator k_n é dado por,

$$k_n = \frac{\omega_n}{c} + i\frac{\delta_n}{c} \tag{5}$$

em que $p_\omega(r)$ é a variação da pressão na posição r e na frequência angular ω, i é a unidade imaginária, $Q = Q(\omega)$ é a função que representa a excitação, ρ é a massa específica do meio de propagação, p_n representa a expansão em autofunção da contribuição da pressão gerada pela fonte, r_0 é a posição inicial da localização do sinal de excitação.

2.1.2 Abordagem de sinais e sistemas

Ao avaliar, o caso particular da Eq. 4, a relação das pressões, dado por $p_\omega(r)$, contidas em dois pontos, r_0 e r, 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 correspondente linear. Para Vorländer e Summers (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 e Summers (2007), é usual expressar excitações como rápidas flutuações finitas localizadas no espaço-tempo com conteúdo espectral amplo. De tal modo, a Eq. (6) apresenta a forma genérica desse tipo de excitação,

$$q(r) = Q\delta(r - r_0) \tag{6}$$

em que $q(r)$ é a excitação, δ é a função Delta de Dirac. Como aludido previamente, a função Q manifesta-se como uma classe de funções harmônicas, e com componentes ortogonais no domínio da frequência. A generalização dessa condição é embutida pela Eq. 7,

$$s(t) = \int_{-\infty}^{\infty} Q(\omega) e^{-i\omega t} d\omega \tag{7}$$

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. 8,

$$p(r,t) = \int_{-\infty}^{\infty} p_{\omega}(r) e^{-i\omega t} d\omega \tag{8}$$

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 planares, ondas 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, como a operação de convolução, entre a entrada e a função de transferência do sistema. A operação de convolução é dada pela Eq. 9,

$$s'(t) = \int_{-\infty}^{\infty} s(\tau) g(t-\tau) d\tau = \int_{-\infty}^{\infty} g(\tau) s(t-\tau) d\tau \tag{9}$$

em que $s'(t)$ é a saída do sistema, τ é uma variável de integração auxiliar, g é a função de convolução. A operação realizada na Eq. 9 é representada na forma compacta como $s'(t) = s(t) * g(t)$, em que $*$ representa o operador de convolução conforme a Eq. 8. Em termos gerais, quando se representa g no domínio da frequência, essa função recebe o nome de função de transferência do sistema.

Cabe salientar o uso dessa modelagem em condições é complexa, 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 a modelagem via os 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 decai 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. 1 (LEHMANN; JOHANSSON; NORDHOLM, 2007), sendo relacionada pela Eq. 10, escrita como,

$$T(\bar{\alpha}) = \frac{0.161V}{\sum_{i=1}^{6} S_i\,\alpha_i} \tag{10}$$

em que T é o tempo de reverberação em segundos, V é o volume da sala [m^3], S_i são as áreas das paredes, correspondentes a parede i-ésima, e α_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 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. 11,

$$E(t) = \int_{0}^{\infty} p^2(\tau)d\tau \tag{11}$$

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 τ é a variável de integração auxiliar. O tempo de reverberação depende da frequência, uma filtragem de sinal, $p(t)$ é feita nas bandas de oitava, 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 remontam do balanceamento de vários fatores. Todavia, uma característica em comum é que estes buscavam estabelecer uma curva padrão da percepção humana subjetiva da fala, ou seja, procuravam 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 usam-se 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 gradua-se a inteligibilidade. Especialmente, nessa classe de descritores subjetivos, a reprodutibilidade dos resultados é função dos indivíduos sob análise, das condições fisiológicas, 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 podem ser empregados 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 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 podem 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 como o padrão de referência internacional para a medição da inteligibilidade no contexto objetivo. De tal modo, tem-se normativas específicas que recomendam valores de referência para o STI. Assim para salas de aula, citam-se a norma Britânica BB93 (DFE, 2015), a norma Finlandesa (SFS 5907, 2004) e a alemã DIN 18041:2004 (DIN, 2004).

2.2.2 O STI indireto

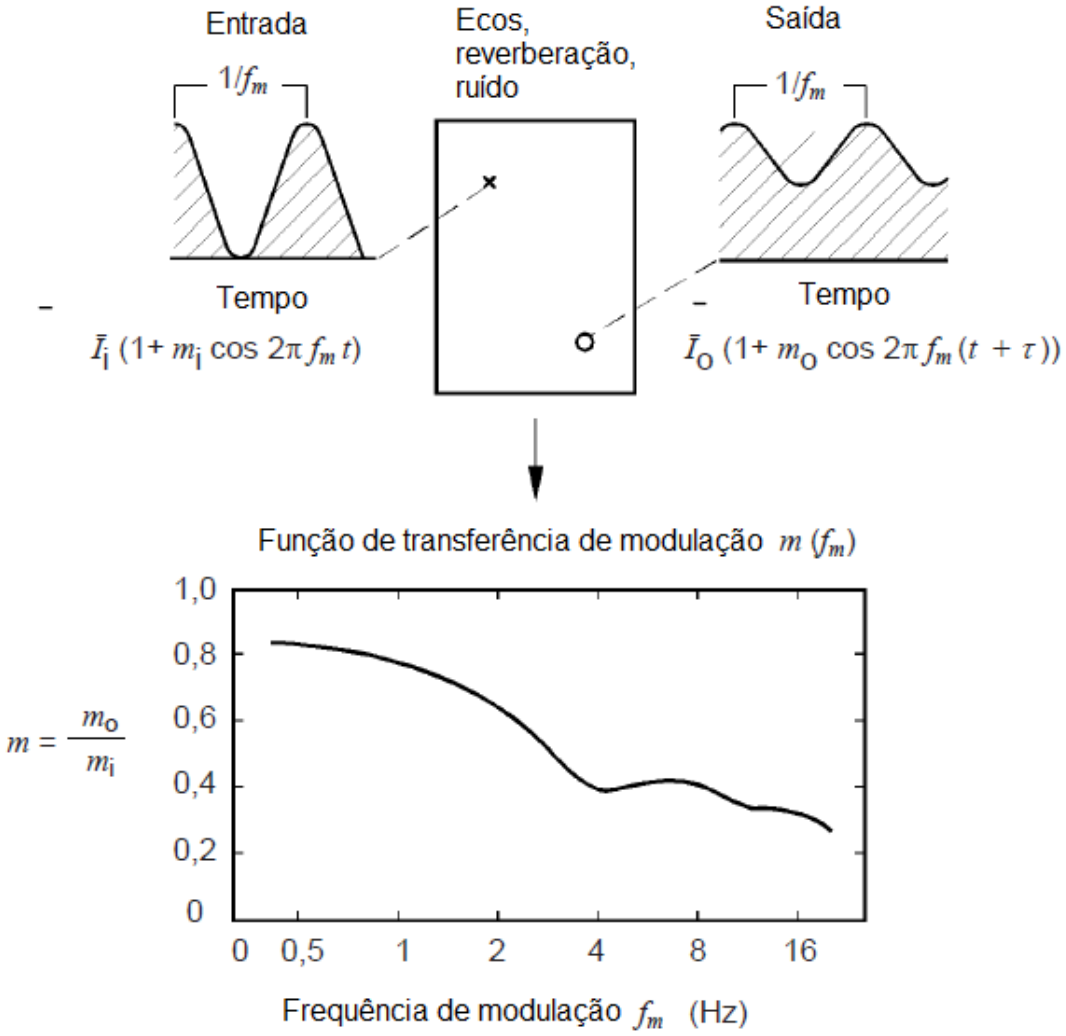
Os procedimentos de medição do STI são estabelecidos na IEC 60268-16 (IEC, 2011). Essa não restringe as condições do STI apenas a salas de aula, mas pode ser aplicada de forma genérica nos sistemas de comunicação amplificados artificialmente, com o por exemplo: em auditórios, sistemas de alerta de segurança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

$$m(F) = \frac{m_0(F)}{m_i(F)} \tag{12}$$

em que $m(F)$ é o fator de redução de modulação na frequência F é a frequência de modulação. Esse fator é usualmente denominado de função de transferência de modulação, $m_i(F)$ 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 evidência 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. 13,

$$m_k(f_m) = \frac{\left|\int_0^\infty h_k^2(t) e^{-j2\pi f_m t}\, d\,t\right|}{\int_0^\infty h_k^2(t)\, d\,t}\left[1 + 10^{-SNR/10}\right]^{-1} \tag{13}$$

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 as frequências de 0,63,

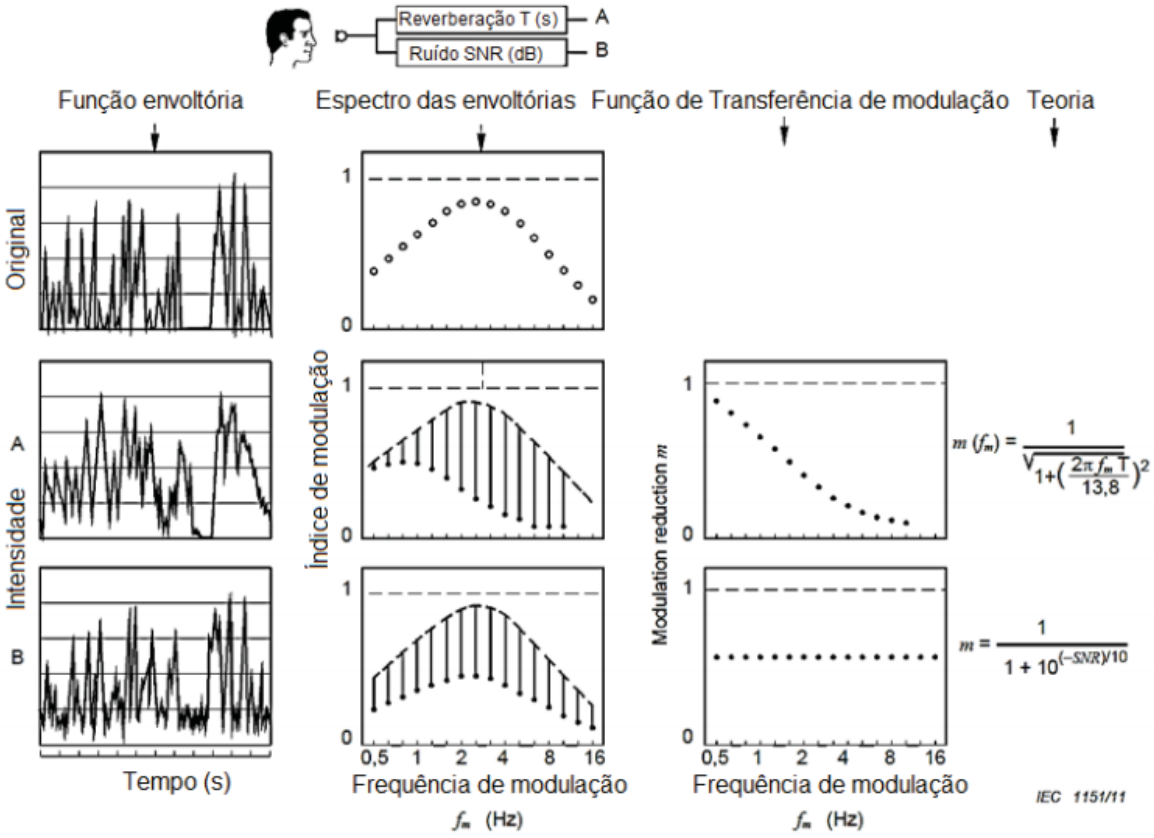
0,80, 1,00, 1,25, 1,60, 2,00, 2,50, 3,15, 4,00, 5,00, 6,30, 8,00, 10,00 e 12,50 Hz, e SNR é a relação sinal-ruído na escala dB, k indica a banda de oitava.

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, 1kHz, 2 kHz e 8 kHz, escrito como,

$$SNR_k = L_{op,k} - L_{BGN,k} \tag{14}$$

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 a 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 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. 15, indica o efeito separado sobre os fatores de modulação.

$$m_k(f_m)_{SNR\to\infty} = \frac{\left|\int_0^\infty h_k^2(t)e^{-j2\pi f_m t}d\,t\right|}{\int_0^\infty h_k^2(t)d\,t} \tag{15}$$

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 dado por,

$$SNR_{eff\,k,f_m} = 10\log_{10}\left(\frac{m(F)}{1-m(F)}\right) \tag{16}$$

em que SNR_{eff} é a relação ruído-sinal efetiva, essa limita-se entre de -15 dB até +15 dB. O resultado dessa normalização é denominado de índice de transmissão, expresso pela Eq. 17,

$$TI_{k,f_m} = \frac{SNR_{eff\,k,f_m} + 15}{30} \tag{17}$$

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

$$MTI_k = \frac{1}{n}\sum_{m=1}^{n} TI_{k,f_m} \tag{18}$$

O valor nominal do STI é,

$$STI = \sum_{k=1}^{7} \alpha_k \, MTI_k - \sum_{k=1}^{6} \beta_k \sqrt{MTI_k \, MTI_{k+1}} \tag{19}$$

em que α_k é o fator de ponderação do sexo no espectro de fala, β_k é fator de redundância. Logo que, 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 mostra as classificações subjetivas dos valores do STI

TABELA 1 – GRADUAÇÃO 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 uma boa qualidade de 0,75 em diante.

2.3 RELAÇÃO ENTRE O RUÍDO, SAÚDE E APRENDIZAGEM

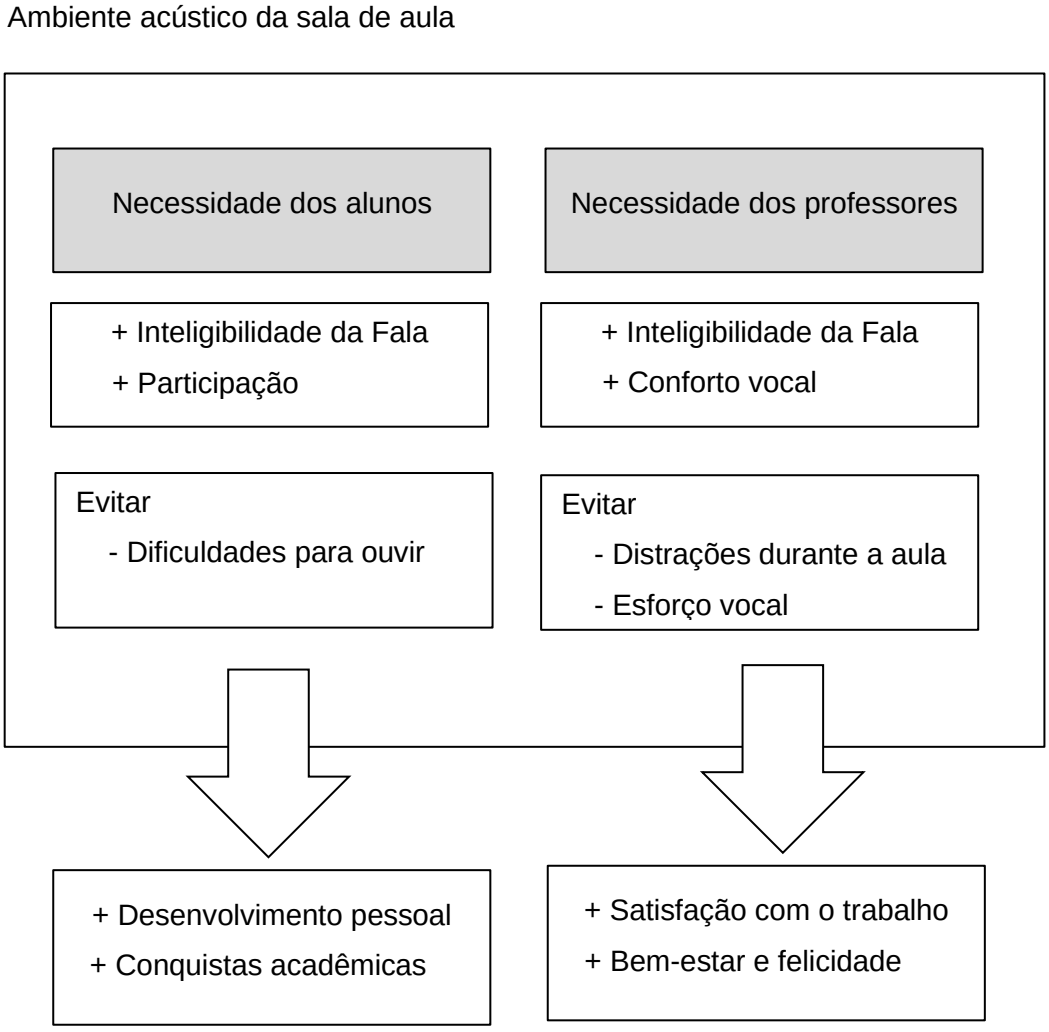
O ruído em contextos escolares para Massonnié et al. (2016) apresenta-se sob duas perspectivas. A primeira em relação a percepção do aluno e a segunda em relação ao professor. Em relação aos professores, destacam-se as decorrências associados à sua atividade profissional, portanto, caracteriza-se uma 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. (2016) mostra que a relação entre o professor e o aluno se estabelece pelo 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 aprendizado ao 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 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 Rasmussen (2018, p, 1074).

Para Rasmussen (2018), em salas de aula a caracterização do ambiente acústico, pode ser expressado 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, o controle de ruído, este compreende-se como 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, tais fontes podem variar conforme a disposição no contexto urbano da localização da sala de aula ou da escola, o que irá propiciar a entrada de ruído, de fontes como o tráfico veicular, movimentação de pessoas, ruídos industriais etc.

Em relação à qualidade acústica, entende-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 as 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 o qual 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, causando 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, C50, EDT, STI dentre outros. Eles identificaram que os professores relatavam 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), geram 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 de grande porte Chinesa. Neste 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) e verificou-se poluição sonora em todas as escolas. Em relação 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 a sua respectiva percepção no ambiente escolar. 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 reconheceram o 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. Sendo que 40% dos entrevistados identificaram que o ruído rodoviário interferia significativamente em suas aulas. 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, 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 tráfico 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 derivavam de situações primariamente internas, como exemplo, o ruído provocado pelos próprios alunos e o ruído transmitido pelas salas próximas.

Ainda, conforme Levandoski e Zannin (2019), a percepção de salas ruidosas foram relatas por 55,7% dos professores. Esta porcentagem foi corroborada pelos altos níveis de ruído relatados de 54,9 dB(A) e 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 ser diretamente relacionado com uma baixa inteligibilidade da fala, portanto, confirma percepção identificada pelos professores.

Notou-se uma recorrência na literatura da manifestação de níveis de ruídos fora dos limites estabelecidos por diversas normativas internacionais. De tal modo, o ruído é um problema comum nos ambientes acadêmicos. Este ainda não se restringe somente há 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, 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, em 9 escolas públicas do estado de Minas Gerais no Brasil. Por meio de

questionários subjetivos 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 ente 0,69 s até 2,09 s, considerando a médias das bandas de oitavas de 500 Hz, 1000 Hz e 2000 Hz. O ruído de fundo variou entre 54 dB(A) até 74 dB(A). Eles concluíram que as contribuições de elevados níveis de ruído de fundo, e altos tempos de reverberação foram onerosos a 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 foi de 0,55 s, e nenhuma sala atingiu os requisitos mínimos. Ainda que, o tempo de reverberação foi relativamente pequeno, a contribuição do ruído de fundo foi significativa para diminuir a qualidade acústica. Quando as salas eram ocupadas pelos alunos, os níveis do ruído de fundo atingiram até 70 dB(A). Como consequência o esforço vocal dos professores nessas foi demasiadamente alto. De maneira geral, os professores tiverem que elevar o tom de suas vozes no mínimo em 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 localizadas em países em zonas tropicais, ou em climas com temperaturas mais elevadas tendem a apresentar sistemas de refrigeração, como ar-condicionado, ventiladores, umidificadores etc. 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 quesito, 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 desliga se a fonte de ruído e faz-se as medições, 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 irão refletir a condição real da sala de aula. Dessa forma, ele propôs avaliar as salas de ambientes universitários concomitantemente as suas atividades de ensino e aprendizagem. Para tanto, medições foram feitas em salas de aula ocupadas e desocupadas. Em seguida, eles 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 ocupados foram menores do que as salas vazias. O maior STI obtido foi de 0,61 ± 0,01. Com isso, Choi (2020) concluí que, medições do STI que emulam as condições reais de aula podem sofrer significativas alterações em relação as salas desocupadas. Assim, inferiu-se que o ruído de fundo do ambiente é um fator primordial (VAN DE POLL et. al., 2014).

Neste momento, 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

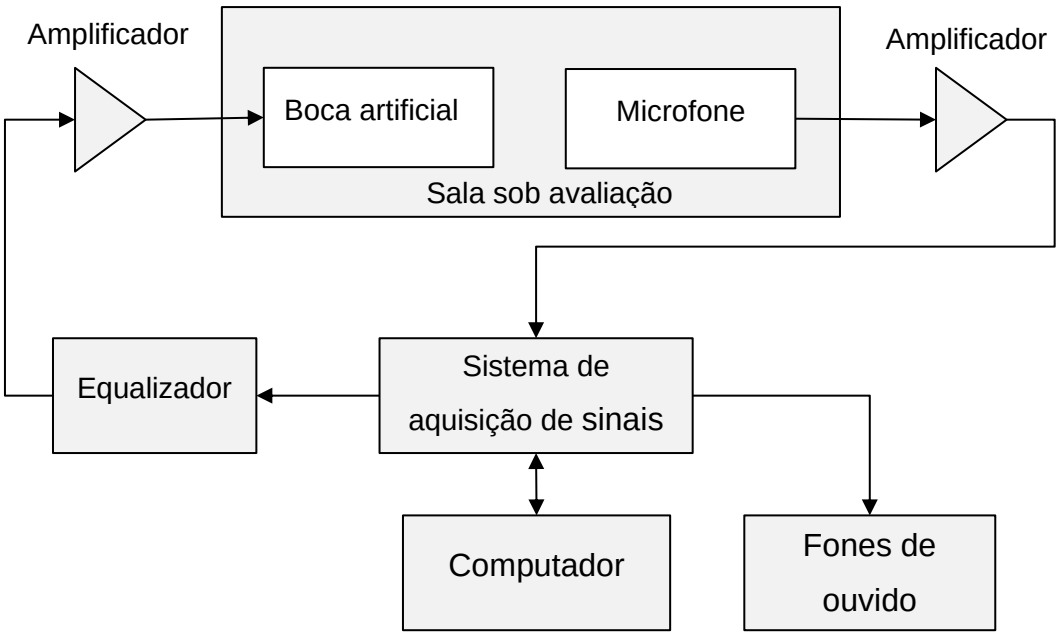
Conforme a IEC 60268-16 (IEC, 2011) há fontes de erros de medição classicamente identificadas, algumas destas fontes incluem, os fenômenos acústicos, como a diretividade da fonte, equalização do sinal, mascaramento, ressonância harmônicas dos modos de vibrar da sala e a não estacionariedade do ruído fundo 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 mais conhecido em inglês como *Just Noticeable Difference* (JND) relativizaram um valor de erro aceitável para o STI. Este conceito no contexto do STI, implica na mínima variação perceptível subjetiva do ouvinte. Dessa forma, os erros de medição devem ser menores que a JND, no caso do STI este é de 0,03. De tal modo, 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 o STI que 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 com um campo difuso, um campo difuso é aquele e que a probabilidade da densidade de energia de uma fonte decair omnidirecionalmente no espaço circunvizinho. Nos campos acústicos não difusos surgem efeitos não lineares. Usualmente, sinais de decaimento harmônicos, ou varreduras senoidais são utilizados como excitação. Todavia, devido a não linearidade e as sobreposições harmônicas essas causam distorções e ressonâncias localizadas no ambiente.

Adicionalmente as 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 seu 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 necessitária de um sistema de amplificação da voz.

Além disso, conforme Liu et al. (2020) com a verificação STI do e sua relação com as medidas subjetivas alteraram-se, motivo pelo qual, 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 e em relação as 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 podem ser

obtidas satisfatoriamente em ruído não estacionário. Para tanto, eles indicaram o requisito mínimo de 25 dB para a relação impulso-ruído. Essa condição restringe-se apenas em excitações do tipo de varredura senoidais. Ainda assim, no mesmo estudo, uma função entre a relação sinal ruído e relação impulso-ruído foi estabelecida. Com isso, em ruído flutuante uma SNR de +15 dB, e INR de +25 dB e um Leq de +75 dB foram os requisitos mínimos globais estabelecidos.

Ainda assim, 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 quesito 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 mostrou-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.

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; LIU et al. 2020). Como consequência, ao aplicar a abordagem tradicional de regressão, a seguir foram destacadas algumas das principais correntes limitações.

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

CARACTER 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 o uso de médias destes como variáveis de regressão;
Qualidade do ajuste baixa	modelos de avaliação com baixo coeficiente de correlação de Pearson;
Interpretação física	limitação referente a interpretação física dos modelos, devido a escala dos descritores;

FONTE: O autor (2023).

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.

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, essa nova expressão ponderava os efeitos do campo reverberante e do ruído de fundo não estacionário. A conclusão deles mostrou que o algoritmo pode prever o STI, tanto em ambientes simulados com em medições in situ, quanto em ambiente 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 os dados de treinamento basearam-se em bancos de dados simuladas das respostas impulsivas e com dados de vozes nítidos que sofreram auralização com as respostas impulsivas das salas. Os áudios foram transformados para 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 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

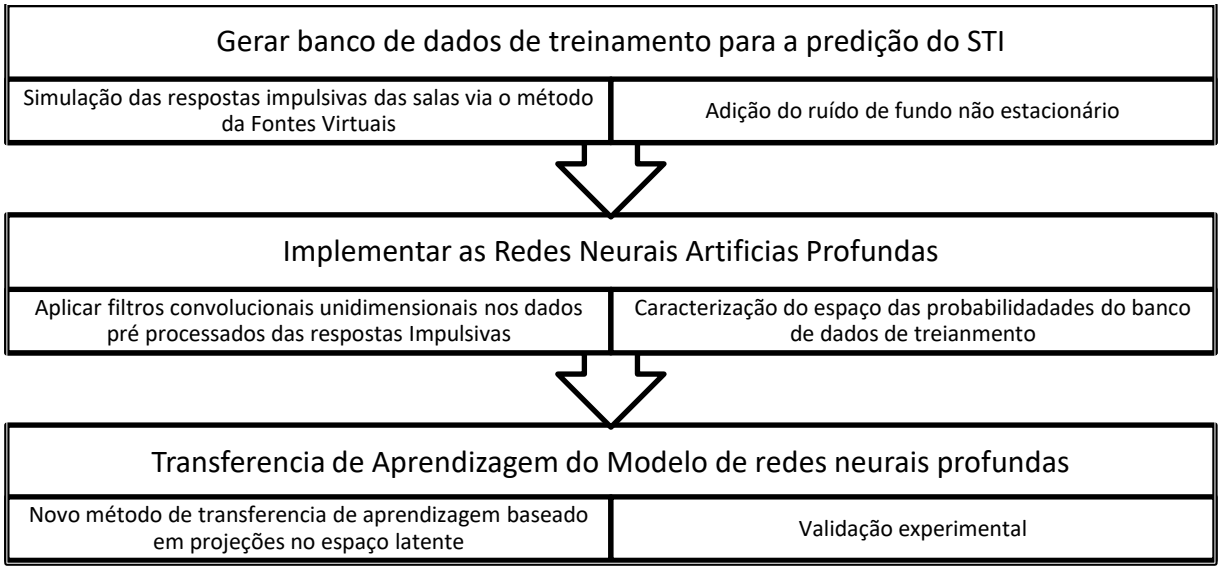
A predição do STI depende de diversos fatores que rotineiramente são avaliados de forma combinada. Dessa forma, verificou-se durante a revisão da literatura uma lacuna nos modelos atuais, tendo em vista que, a escolha arbitrária das variáveis de regressão pode gerar modelos que são de difícil reprodutibilidade devido à pouca quantidade amostral.

Diante disso, essa sessão 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 com 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 possiblidade da integração de medições experimentais ao modelo. A FIGURA 5 mostra a metodologia proposta neste trabalho.

FIGURA 5 – ESQUEMA METODOLÓGICO PROPOSTO



FONTE: O próprio autor (2023).

O uso de redes neurais artificiais como modelos não paramétricos para aproximação funcional, possuem 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 forneçam 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 Schoreder. este modelo determina os fatores de redução de modulação. Neste, os efeitos do ruído de fundo e do tempo de reverberação são independentes, similar 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.

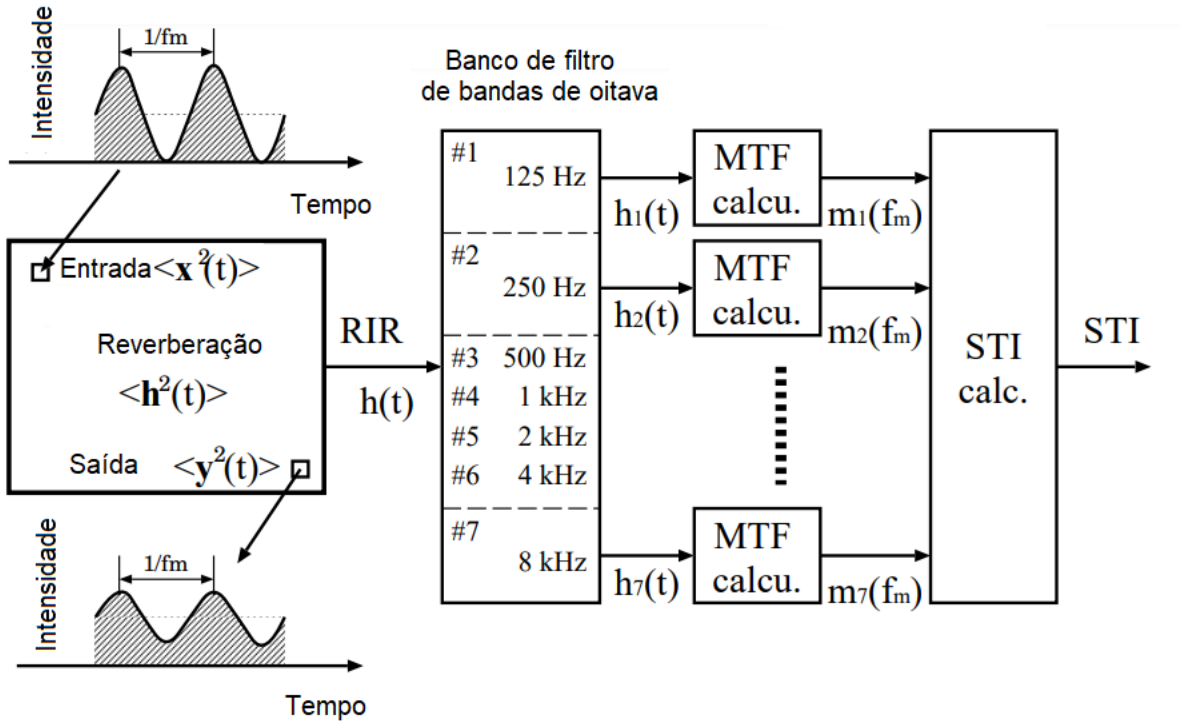
Enquanto, na medição indireta, o efeito da reverberação é dado pela excitação com um nível operacional que emula o espectro da voz masculina, e seu nível é padronizado para 60 dB(A). Uma vez que se emula o espectro da voz masculina o fator de direcionalidade da fonte deve ser considerado como não direcional. De posse das respostas impulsivas e do ruído de obtém-se os fatores de redução de modulação m via Eq. 20,

$$m(f_m) \cong \left\{1 + \left(2\pi \frac{T}{13.8}\right)^2\right\}^{-1} \left[1 + 10^{-SNR/10}\right]^{-1} \quad (20)$$

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, SNR é a relação ruído sinal. A obtenção é STI uma vez determinados os fatores de produção foram descritos 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, similar 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 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 dado 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.

Ao findar esta seção, obteve-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

Como 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 experimentais foi a criação de uma base de dados. Essa base foi composta 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). Com isso, após utilizou-se a metodologia de transferência de aprendizagem para otimizar a generalização dos resultados dos treinamentos das redes neurais artificiais profundas, como base nas medições *in situ*, tanto para o STI quanto o TR.

O método das imagens, mais conhecido em inglês como*, Image Source Method* (ISM) foi empregado para obter os sinais simulados no domínio do tempo das respostas impulsivas. O ISM considera a fonte num recinto, dada pela Eq. 21,

$$P(\omega, X, X') = \frac{e^{i\omega(R/c-t)}}{4\pi R} \tag{21}$$

em que P é pressão sonora, X e X' representam a posição da fonte e do receptor respectivamente, R é o vetor distância entre X e X'. Onde 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,

$$P(t, X, X') = \sum_{p=1}^{8} \sum_{r=-\infty}^{\infty} \frac{\delta\left[t - \left(\left|R_p + R_T\right|\right)/c\right]}{4\pi\left|R_p + R_T\right|} \tag{22}$$

em que P é pressão advinda da resposta impulsiva simulada, δ é o Delta de Dirac, R_p e R_T são as funções de reflexão especular. 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 desse 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 10000 salas retangulares virtuais foram projetadas com dimensões aleatórias como o domínio fechado, 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 as 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 e foi atribuído de uma distribuição gaussiana no início de cada simulação, até obter 10000 respostas impulsivas. Os coeficientes de absorção sonora do som correspondentes foram obtidos resolvendo a Eq. 10 para o coeficiente de absorção sonora.

Como explicitado previamente, as amostras dos sinais que emularam o ruído de fundo foram derivadas do 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 do 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 do ruído de fundo. Pós processou-se estes utilizando a densidade espectral, ou densidade espectral de potência, dado por,

$$S_{xx}(\omega) \int_{-\infty}^{\infty} |h(t)|^2 dt = \int_{-\infty}^{\infty} |H(\omega)|^2 \, d\omega \tag{23}$$

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 do periodogramas de Welch (WELCH, 1967). 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. Além disso, na formulação do STI indireto via Equação de Schroeder (SCHROEDER, 1981). 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 pré-processamento mantém-se 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.

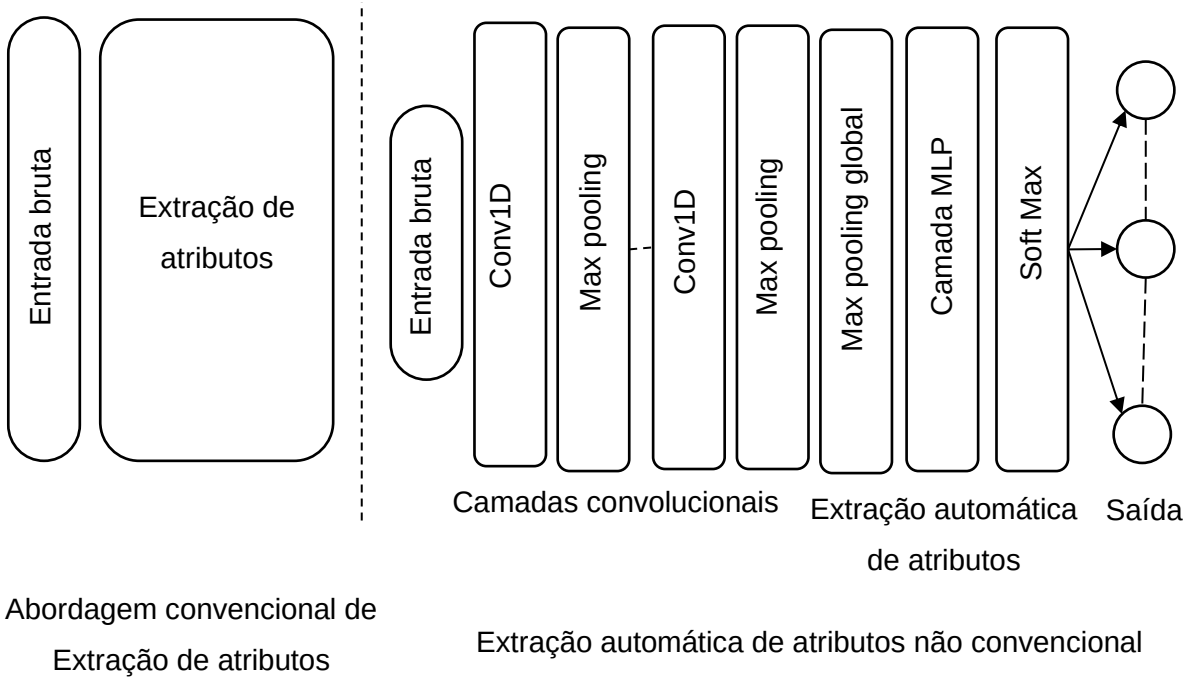
Por conseguinte, as redes neurais profundas em séries temporais usando a camada de Rede Neural de Convolução unidimensional (Conv1D) são empregadas como técnicas de extração de atributos dos sinais de entrada. Neste trabalho, adotou-se o viés da abordagem de engenharia de atributos como o equivalente 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, 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, sendo a extração de características tradicional 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 de sem pré-processamento. Consequentemente, a camada Conv1D aumenta a precisão em sistemas de reconhecimento de voz automático (PALAZ; MAGIMAI-DOSS; COLLOBERT, 2019).

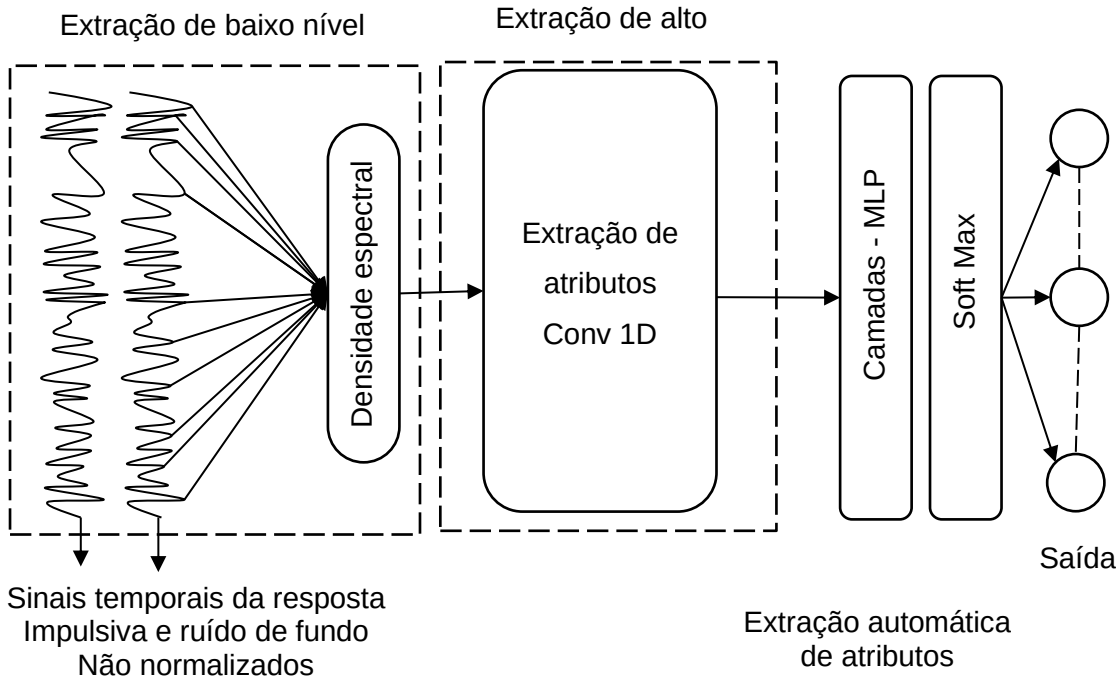
Além disso, tal abordagem como afirmaram Mon, Pa-Pa e Thu (2018) possui alguns benefícios em relação à redução da complexidade do modelo. Logo, um modelo parcimonioso pode atingir uma precisão de última geração quando comparado com as abordagens tradicionais de extração de recursos, como a Transformada 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 define-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 um alvo ordenados. As entradas foram a densidade espectral das respostas impulsiva 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. Nessa verifica-se que a resposta impulsiva e o ruído de fundo no domínio do tempo são transformados para o domínio da frequência via a densidade espectral. Em relação ao conjunto de treinamento, os dados de entrada foram normalizados aplicando-se a transformação via z-score, conforme a Eq. 24,

$$x'_{m,i} = \frac{x_{m,i} - \mu(x_{m,i})}{\sigma(x_{m,i})} \tag{24}$$

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 a densidade espectral de potência da resposta impulsiva da sala e $m = 2$ identifica a densidade espectral de potência do ruído de fundo, $s = 10000$ é a quantidade total de amostras de treinamento, $x_{m,i}$ 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. 25,

$$\mathcal{X} = (X, Y)_{s*b,j+k} = \left(\{x_{m,i,b}\}_{i=1}^{j}, \{y_{m,i,b}\}_{i=1}^{k}\right)_{m=1}^{s} \tag{25}$$

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, que quando expandida resulta na Eq. 26,

$$\{x_{m,i}\}_{i=1}^{s} = (x_{11}, x_{12}), \dots, (x_{12}, x_{m,i}), \tag{26}$$

em que é a $\{x_{m,i}\}_{i=1}^{j=2}$ 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 20, 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, definida na seção 3.3.3.

O critério da adoção do número de classes, $k = 20$, 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, inferiu-se uma extrapolação para definir a quantidade de classes, uma vez que, com o número de classes, $k = 20$, a diferença entre as classes é de 0,05, 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 Conv1D

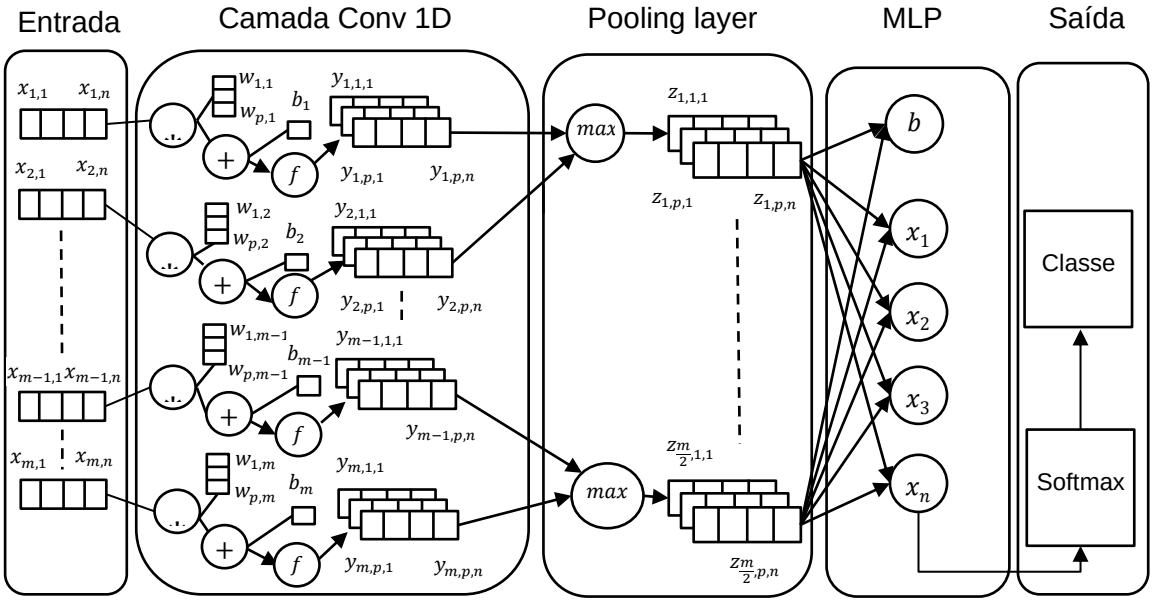
Após a extração das características de baixo nível realizada na seção 3.3.1, o sinal resultante foi direcionado para a primeira camada de entrada. Essa camada constitui o alto nível de extração de atributos de uma rede neural convolucional unidimensional. A convolução discreta aplica nessa camada é,

$$h_i(n) = (x * w)(n) = \sum_{j=1}^{m} w(j)x(i-k) \tag{27}$$

em que $*$ é a operação de convolução, i é o número dos filtros de saída, onde w vetor é o kernel com comprimento m é o número de filtros, $i-k$ é o comprimento da janela e x 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



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.

$$y_{i,j,k} = \sigma\left(\sum_{i=1}^{s} x_{i,k} * w_{j,i} + b_i\right) \tag{28}$$

em que $y_{i,j,k}$ é o mapeamento de saída da rede neural convolucional unidimensional, σ 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 $x_{i,k}$ e $1 \leq i \leq m$ e b_i representa o termo de viés. As faixas de variação de j e k são, $1 \leq j \leq m$ 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, 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*, esse 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.

$$z_{i,j,k} = max_{k=1}^{r}\left(y_{i-1,j,k}\right) \tag{29}$$

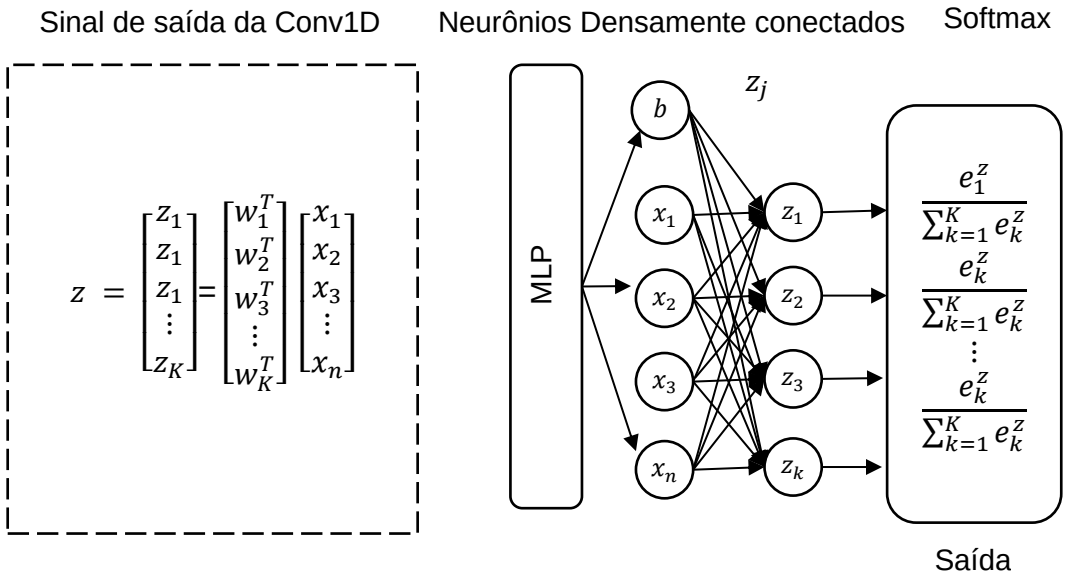
em que $z_{i,j,k}$ é a saída máxima da camada *max-polling*, $y_{i-1,j,k}$ é o peso da camada anterior, max é o operador matemático máximo. Esse tipo de camada aumenta o desempenho do treinamento, pois os efeitos de pequenos pesos são minorados da otimização do gradiente (PAN et al., 2018).

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 Sofmax, esse procedimento será tratado na seção 3.3.3

3.3.3 Ajustes dos pesos

As atualizações iterativas dos pesos na camada Conv1D serão 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 Singh (2020, p. 444).

Sem a perda de generalidade, na função de saída SoftMax na FIGURA 10, adotou-se o parâmetro θ 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. Essa retorna a probabilidade condicional, dado um conjunto de pesos, na camada densamente conectada, e estima-se a pertinência, ou seja, a probabilidade da amostra de entrada pertencer a classe de referência, conforme mostrado na Eq. 30

$$h_\theta\left(x^{(i)}\right) = \begin{bmatrix} p\left(y^{(i)} = 1|x^{(i)};\theta\right) \\ p\left(y^{(i)} = 2|x^{(i)};\theta\right) \\ \vdots \\ p\left(y^{(i)} = 1|x^{(i)};\theta\right) \end{bmatrix} = \frac{1}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_l^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix} \quad (30)$$

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 como o 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 retro propagação dos erros, aplicando a abordagem dos filtros convolucionais. Concomitantemente, Kiranya, Ince e Gabbouj (2015) e Wang et al. (2019) consolidaram um modelo de otimização via gradiente descendente estocástico para o ajuste dos pesos, neste 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). Enquanto, que a atualização dos pesos na camada densamente conectada seguiu a abordagem tradicional de retro propagação 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. 31 (ZENG et al., 2014),

$$w_{i,j} = \sum_{j}^{s} w_{j,i}^{l-1} \sigma\left(z_{i,j,K}\right) + w_i^{l-1} \quad (31)$$

em que a interação dos pesos é,

$$w_{i,j} = w_{j,i}^{l-1} - \alpha \frac{\partial \mathcal{L}_i}{\partial w_{i,j}} \quad (32)$$

em que o gradiente é dado como,

$$\frac{\partial \mathcal{L}_i}{\partial w_{i,j}} = \sum_{i=1}^{C} t_i \frac{\partial \mathcal{L}_i}{\partial w_{i,j}} \tag{33}$$

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, $\mathcal{L}$ 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. 31, (KINGMA; BA, 2014, CHOLLE et al., 2018),

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C} \mathbf{1}(y_i) \frac{e^{w_k^T b_i}}{\sum_{j=1}^{C} e^{w_k^T b_i}} \tag{34}$$

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, 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 do peso (CHOLLE et al., 2018). O termo Adam não é um acrônimo, 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 a este é apresentado 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 artificias profundas aplica-se o método da validação cruzada, k-cross validation, para avaliar a qualidade do estimador. Para tanto, neste trabalho adotou-se dois conjuntos, o 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 presente trabalho, empregou-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 do acurácia, ACC, e o $F1-score$. A acurácia é dada pela Eq. 35,

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{35}$$

em que ACC é a acurácia do modelo, esse parâmetro representa o desempenho global, TP é a quantidade de positivos verdadeiros, TN é a quantidade de falsos negativos, FP é a quantidade de falso positivos e FN é a quantidade de falso negativos. O F1 score é particularmente útil pois informa o balanço entre a acurácia e a frequência de acerto e uma determinada classe. O $F1-score$ é dado por,

$$F1-score = 2 \times \frac{\frac{TP}{TP+FP}\frac{TP}{TP+FN}}{\frac{TP}{TP+FP}+\frac{TP}{TP+FN}} \tag{36}$$

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, empregou-se a 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 $\mathcal{A}$, este modelo possui os hiperparâmetros, λ. Logo, identifica-se o modelo hiper-parametrizado 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 λ^* que resultam no modelo de referência, denotado por $\mathcal{D}$ em que minimiza a diferença entre o melhor modelo possível e o modelo obtido. A estimativa do modelo ótimo é

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \mathbb{E}_{(D_{treino}, D_{validação}) \sim D} \mathcal{L}(\mathcal{A}_\lambda, D_{treino},) D_{validação} \tag{37}$$

em que λ^* representa o conjunto ótimo de hiperparametros, $\mathcal{L}(\mathcal{A}_\lambda, D_{treino},) D_{validação}$ é a função objetivo do modelo de aprendizagem profunda do modelo $\mathcal{A}$, ver Eq. 33. 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 adotara a abordagem da Eq. 36, o problema corresponde otimização da acurácia dos modelos. Como corolário, realiza-se a maximização das métricas de ajuste da qualidade do modelo, conforme as Eq. 34 e Eq. 35. De tal modo, a TABELA 3 mostra as topologias avaliadas para o emprego da otimização de hiperparâmetros.

TABELA 3 – TOPOLOGIAS E HIPERPARÂMETROS DOS MODELOS DE REDES NEURAIS

Camada	Nome da camada	Função de Ativação	Hiper parâmetro
1	Camada de entrada	#	Tamanho do lote: 32, 64 e 128
2	Conv1D	ReLu	Tamanho do kernel: 2, 8 e 16 Número de filtros: 2, 8, 16
3	Conv1D	ReLu	Tamanho do kernel: 2, 8 e 16 Número de filtros: 2, 8, 16
4	MaxPooling1D	#	#
5	Conv1D	ReLu	Tamanho do kernel: 2, 8 e 16 Número de filtros: 2, 8, 16
6	Conv1D	ReLu	Tamanho do kernel: 2, 8 e 16 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	ReLu	Número de neurônios: 10; 100 e 200
10	Camada MLP	ReLu	Número de neurônios: 10; 100 e 200
11	Saída do FC	Sofmax	20

FONTE: O autor (2023).

Na TABELA 3 observa-se a alocação de camadas duplas, tanto para as camadas de Conv1D e Camada MLP, isso deve-se, ao aumento intrínseco da capacidade de generalização (ZHA 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. Os espaços latentes são denominações genéricas de espaços tensoriais, vetoriais e subespaços dos pesos sinápticos que são aproximáveis.

Com base na otimização dos hiperparâmetros realizou-se uma busca de malha uniforme para identificar a melhor combinação destes que minimiza a Eq.36. Adicionalmente, usou-se a média de 10 treinamentos de cada modelo de forma independente e avaliou-se os resultados das métricas de qualidade, no caso, a acurácia ACC e o fator $F1 - score$.

Abordagens similares a 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 APRENDIZAGEM POR TRANSFERÊNCIA

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., 2020).

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. Conquanto, existam aplicações

de engenharia onde 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 (HARRIS, 1991, LI; GRANDVALE; DAVOINE, 2020). Quando tais casos emergem é mais adequado desenvolver outra heurística de treinamento, que cubra essa lacuna na aquisição de amostras de treino.

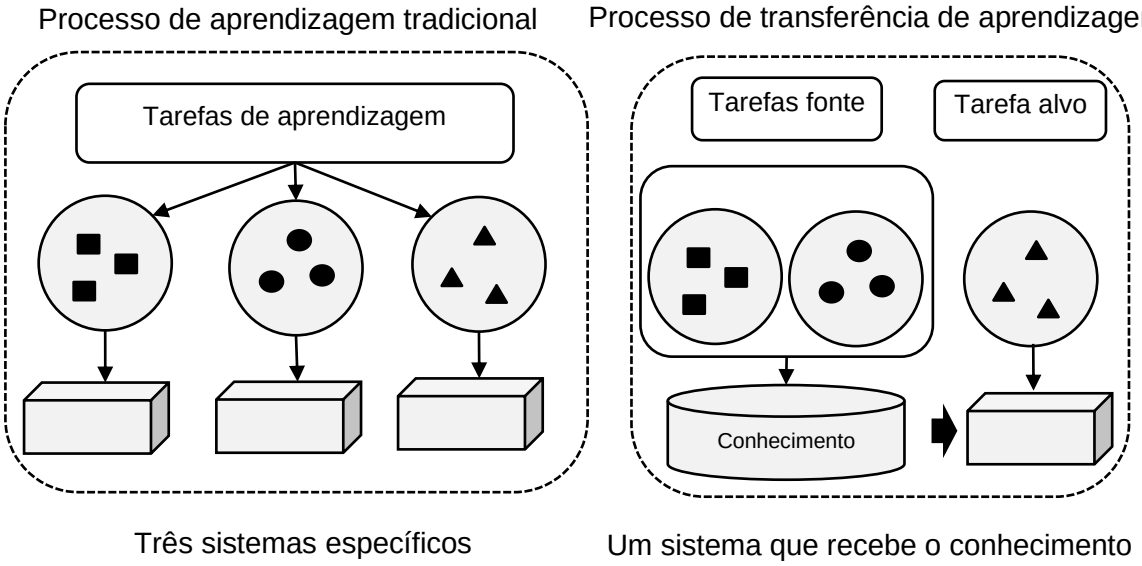
O fenômeno previamente mencionado é especialmente verdadeiro no campo da acústica, especialmente quando há fortes não linearidades. Quando a saída de um sistema pode se comportar de maneira completamente diferente, dadas pequenas variações em suas condições iniciais e de contorno (SAARELMA et al., 2020). Por continuidade, ainda mesmo, em casos linearizados as funções como coerência e correlação são normalmente usadas para identificar dentro de um determinado sistema linear invariante no tempo, a resposta quando esse sistema é excitado em locais diferentes.

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 as 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 destino. 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 a estabelecer a transferência de aprendizagem, define-se dois domínios representados por espaços vetoriais. A FIGURA 10 mostra a comparação entre as abordagens tradicionais e a transferência de aprendizagem.

FIGURA 11 – DIFERENCIAÇÃO DA APREDIZAGEM DE MÁQUINA TRADICIONAL E DA TRANSFERENCIA DE APRENDIZAGEM



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

Observa-se na FIGURA 10(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 10(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.

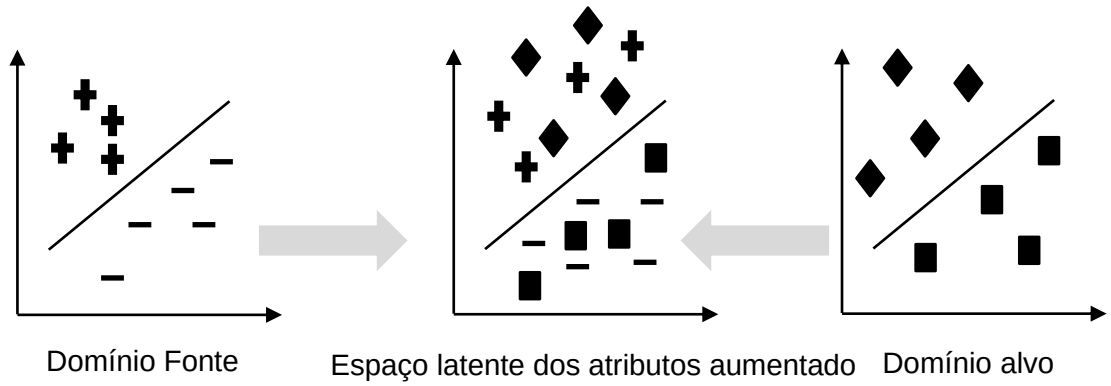
Nota-se que transferência pode ser dada pelo uso dos pesos sinápticos previamente ajustados. Normalmente, em aplicações práticas, a escassez de dados pode emergir, limitando o uso dos modelos de aprendizagem profunda. Nesses casos, a transferência de aprendizagem é particularmente eficiente.

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 $\mathcal{X}$. Enquanto, 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 respetiva função de distribuição condicional probabilística, que foi aprendida pelo

classificador. No contexto deste trabalho, $\not{f}$ 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 (2020, 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,

$$\not{f}(x_j) = \{\mathcal{P}(\mathcal{y}_{\not{k}}|x_j)|\mathcal{y}_{\not{k}} \in |\mathcal{Y}, \mathcal{K} = 1, \cdots, |\mathcal{Y}|\} \tag{38}$$

De tal 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. 39 mostra a função objetivo genérica aplicada na transferência de aprendizagem (POELITZ, 2014),

$$\mathcal{L}\big(p_s(x), p_t(x)\big) = D(p_s, p_t) + \lambda d(h_s, h_t) \quad (39)$$

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, λ é um parâmetro regularizador e d estima a distância entre as duas distribuições, a qual realiza 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 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 APRENDIZAGEM POR TRANSFERÊNCIA

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

3.5.1 Transferência de aprendizagem baseado 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. 38. 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 destino deve coexistir sob alguma extensão

mensurável. Para tanto, uma variante da Eq. 39 surge, essa pondera o efeito do distanciamento no espaço das distribuições, do mesmo modo, o efeito da variância das classes é considerado simultaneamente, conforme,

$$\min_{\Phi} \frac{\big(DIST(\mathcal{D},\mathcal{T};\Phi) + \lambda\Omega(\Phi)\big)}{VAR(\mathcal{D}\cup\mathcal{T};\Phi)} \tag{40}$$

em que Φ 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. 39 e a Eq. 40, revela que ao adaptar uma abordagem de projeção em baixas dimensões, ou ao utilizar espaços latentes pode-se obter uma descrição estatísticas robustas dos dados.

No entanto, deve-se atentar quando a não afinidade surge entre as distribuições marginais dos domínios de origem e de alvo. Para lidar com essa questão, a métrica denominada de Máxima Discrepância Média, em inglês, *Maximum Mean Discrepancy* (MMD). A Eq. 41 foi uma das primeiras tentativas de quantificar de uma maneira semiempírica um método de distribuição-distância para expressar a similaridade estatística das amostras provenientes do sub espaço de treinamento.

$$MMD(X^s, X^T) = \left\| \frac{1}{n^S}\sum_{i=1}^{n^S}\Phi\big(x_i^S\big) - \frac{1}{n^T}\sum_{j=1}^{n^T}\Phi(x_i^T) - \right\|_{\mathcal{H}}^2 \tag{41}$$

em que n^S e n^T são as quantidades de amostras nos domínios da fonte e alvo respectivamente, $\Phi(\cdot)$ 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, $\|\cdot\|$ 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. As projeções em estados próprios, espaço próprio não linear e expansões de kernel o método amplamente utilizado. Em relação aos métodos tradicionais Zheng et al. (2019) e Zhuang et al. (2019) estratificaram uma divisão do sentido de aprendizagem baseada em projeção. Neste eles consolidaram os métodos de Análise de Componentes Principais do Kernel (KPCA), o autoencoder variacional (VAE), além da Máxima Discrepância Média (MMD).

3.5.2 Análise de Componentes Principais por Núcleo

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). Wibowo (2018) propõe a seguinte 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 Ψ aplicada em $\{x_i\}_{i=1}^{N}$. Essa realiza uma transformação para uma dimensão superior no espaço dos atributos. O espaço dos atributos é representado por $\mathcal{F}$, tal que, Ψ: $\mathbb{R}^p \rightarrow \mathcal{F}$. A transformação equivalente é dada por,

$$\Psi = (\Psi(x_1)\ \Psi(x_2)\ \cdots\ \Psi(x_N)\ )^T \tag{42}$$

em que Ψ 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,

$$\hat{C} = \frac{1}{N}\sum_{i=1}^{N} \Psi\,(x_i)\Psi(x_i)^T = \frac{1}{N}\Psi^T\Psi \tag{43}$$

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

Originalmente, Rosipal e Trejo (2001) mostraram que a forma de Ψ não precisa ser necessariamente conhecida. Para tanto, emprega-se um artifício matemático denominado de Truque do Kernel. Este truque permite a diagonalização de $\mathbb{k}$, por meio

da decomposição em autovalores do problema equivalente de $\lambda u = \hat{\mathbb{C}}u$, onde u representam os autovetores e λ autovalores de $\hat{\mathbb{C}}$. A decomposição via matriz kernel fica definida como,

$$k = \Psi^T \Psi \tag{44}$$

a matriz kernel, k, possui dimensão de $N \times N$, em que o problema se transforma para,

$$k\tilde{u} = n\lambda\tilde{u} \tag{45}$$

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

$$u^k = (n\lambda_k)^{-1/2}\Psi^T\tilde{u}^k = \tilde{\lambda}_k\Psi^T\tilde{u}^k \tag{46}$$

em que $\tilde{u}^k$ é a estimativa do autovalor obtido pela solução da Eq. 45. A projeção do autovetor ortogonal sobre os dados originais é dada por,

$$\beta_k(x) = \Psi(x_i)^T u^k = \tilde{\lambda}_k^{\ -1/2} \sum_{i=1}^{n} u_i^k K(x_i, x) \tag{47}$$

Para Shawe-taylor e Cristianini (2004) em implementações computacionais, a solução padrão do problema de autovetores numérica pode ser aplica da diretamente na Eq. 45. 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 apresentas na seção 3.6.

3.5.3 Redução dimensional via autoencoders variacionais

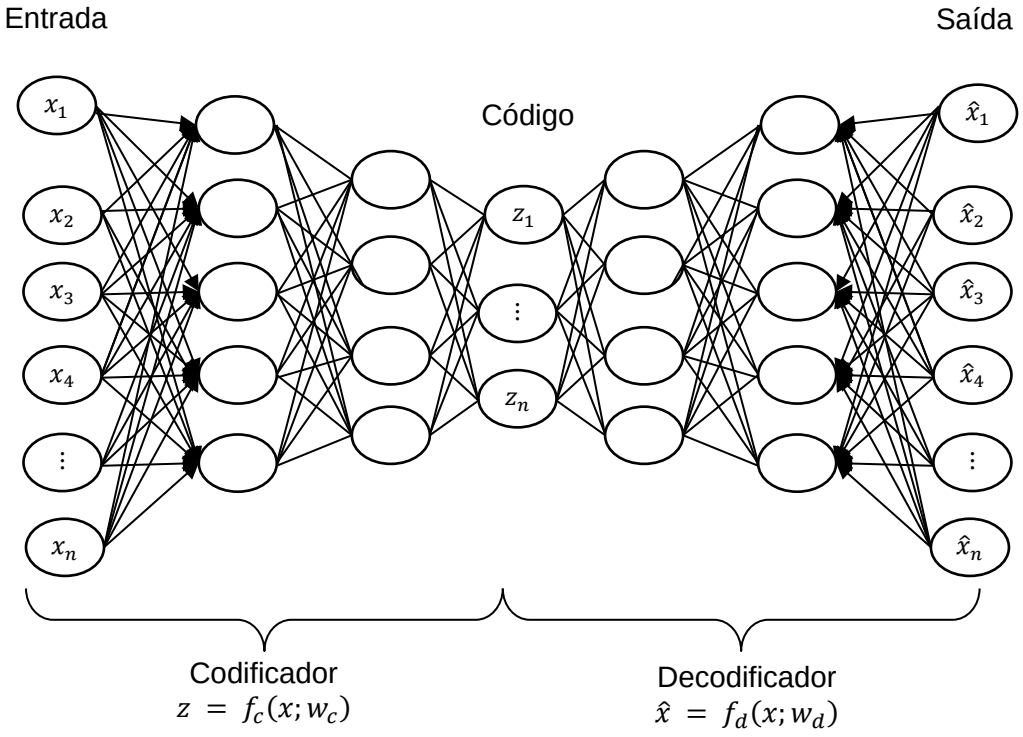
Na seção 3.4.1, segundo Zheng et al. (2019) as funções bayesianas não necessariamente são ótimas candidatas para os problemas de transferência de aprendizagem. Por outro lado, Rezende, Mohamed e Wierstra (2014) aludiram que a formulação híbrida entre o método variacional bayesiano e as redes neurais podem

ser particularmente úteis. Essa utilidade é dada pela transformação de um problema inferencial estatístico multidimensional em um problema de otimização numérico convexo.

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 auto-codificadoras, 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 auto-codificadora.

FIGURA 13 – REDE NEURAL AUTO CODIFICADORA PADRÃO



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

A FIGURA 13 mostra que, esse tipo de auto codificadora possui três elementos, o codificador, o código e o decodificador. O mapeamento entrada-saída é dado por, x→x̃, 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 o 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.

Patterson e White (2018) argumentaram que nesse estrangulamento gera-se um tipo de generalização, inspirado no mapeamento por estruturas isomorfas bijetoras, mas uma classe especial denominada de Auto Codificadores Variacionais, ou em inglês, *Variational autoencoders* (VAE) são mais robustos. Portanto, o VAE é uma função análoga de um mapeamento bem definido. Para Canchumuni, Emerick e Pacheco (2019) o caráter bayesiano surge no momento da estimativa da distância entre a distribuições, $\mathrm{p(z|x)}\|\mathrm{p(x)}$. A formulação do problema de otimização fica definido como,

$$\mathcal{L}(x) = \mathcal{L}_{RE}(x) + \mathcal{D}_{KL}(p(z|x) \parallel p(z)) \tag{48}$$

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

$$\mathcal{L}_{RE}(x) = -\frac{1}{N_x}\sum_{i=1}^{N_x}[x_i\, ln(\hat{x}_i) + (1-x_i)\, ln(1-\hat{x}_i)] \tag{49}$$

para $\mathcal{L}_{RE}(\mathrm{x})$ considerou-se a função entropia cruzada binária. Para essa função, as classes k dos dados foram representados pela notação de codificação one-hot, ver seção 3.3.1.

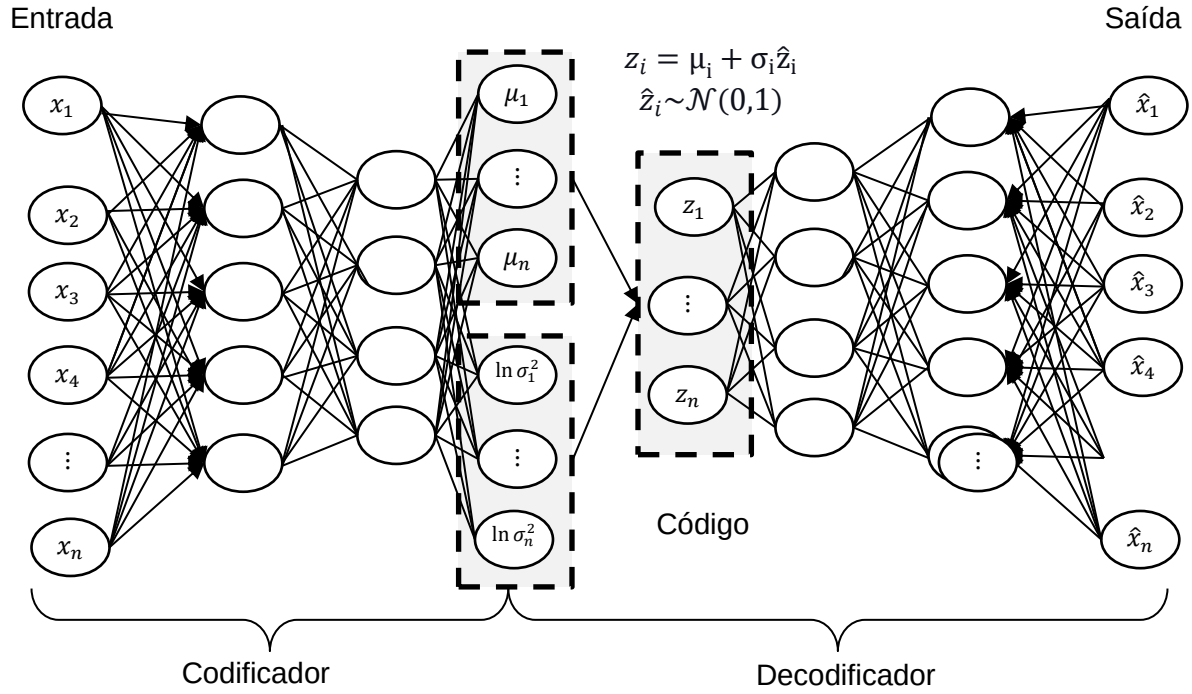
$$\mathcal{D}_{KL}(p(z|x) \parallel p(z)) = \frac{1}{2}\sum_{i=1}^{N_z}(\mu_i^2 + \sigma_i^2 - ln(\sigma_i^2) - 1) \tag{50}$$

em que $\mathcal{D}_{KL}$ é a divergência de Kullback-Leibler, μ_i e σ_i é a s média e o desvio padrão das amostras respectivamente.

O cerne do VAE incide sobre a redução dimensional não linear do espaço dos atributos para z ao adicionar mais uma camada que é responsável pela caracterização estatística da amostra x. Portanto, a minimização é feita por método de gradiente estocástico bayesianos. 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, $z_i = \mu_i + \sigma_i \hat{z}_i$, onde, $\hat{z}_i \sim \mathcal{N}(0,1)$, em que $\mathcal{N}$ representa uma função densidade de probabilidade gaussiana. A gaussiana, $\mathcal{N}(0,1)$, 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 AUTO CODIFICADORA VARIACIONAL



FONTE: Adaptado de Weiss, Khoshgoftaar e Wang (2020, p. 88).

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. Devido a isso, essa capacidade do VAE é abordada na seção 3.6 à qual delineará um novo paradigma de aprendizagem.

3.6 ALGORITMO PROPOSTO

O algoritmo proposto 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 deste é realizar uma busca de malha aleatória sobre o espaço latente gerado pelo VAE. O espaço latente resultante é obtido pela parametrização pela 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 via o Algoritmo da Evolução Diferencial (STORN; PRICE, 1997).

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 e o transforma em um problema de autovalores não lineares por meio do KPCA. Em seguida, deflaciona-se a matriz kernel centralizada no espaço dos atributos e obtém uma aproximação não linear. Dessa forma, referindo a notação da seção 3.5.2, o KPCR fica definido como,

$$y = \Psi\xi + \epsilon \tag{51}$$

em que y é um vetor com N observações, Ψ é a matriz de mapeamento no espaço dos atributos, ξ é o vetor dos coeficientes de regressão, $\Psi\colon \mathbb{R}^p \to \mathcal{F}$, que é aplicada sobre $\Psi(x_i)$, onde $\{x_i\}_{i=1}^N$, ϵ 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, essa é proporcional a $\Psi^T\Psi$. Com isso, projeta-se os autovalores, ver Eq. 47, sobre espaço dos atributos e obtém-se a estimativa de Eq. 51, como,

$$y = Bw + \epsilon \tag{51}$$

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

$$V = \sum_{i=1}^{n} \alpha_i \Psi(x_i) \tag{53}$$

soluciona-se a regressão com base no estimador, $\widehat{w}$,

$$\widehat{w} = (B^T B)^{-1} B^T y = \Lambda^{-1} B^T y \tag{54}$$

em que Λ é a matriz espectral deflacionada em relação a 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,

$$f(x, c) = \sum_{k=1}^{p} w_k \tilde{\lambda}_k^{-1/2} \sum_{i=1}^{n} \alpha_i^k K(x_i, x) + b \tag{55}$$

o que é equivalente a,

$$f(x, c) = \sum_{i=1}^{n} c_i K(x_i, x) + b \tag{56}$$

onde, c_i é:

$$c_i = \left\{ \sum_{k=1}^{p} w_k \, \alpha_i^k \right\}_{i=1}^{n} \tag{57}$$

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, Eq. 58 é similar ao codificador desenvolvido da rede VAE e essa classe de funções são utilizadas nas redes neurais da base radial (LI et al, 2020).

$$K(i,j) = \Psi(x_i)^T \Psi(x_j) = e^{\left(\frac{\|x_i - x_j\|}{2\delta^2}\right)} \tag{58}$$

em que $K(i,j)$ é a matriz kernel, δ é um hiper parâmetro que informa a largura de banca da gaussiana, $\|\cdot\|$ é a norma euclidiana.

Logo que, definido 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,

$$\widetilde{K} = K - 1_N K - K 1_N + 1_N K 1_N \tag{59}$$

em que é $\widetilde{K}$ matriz kernel centralizada no espaço dos atributos, 1_N é a matriz unitária quadrada com dimensão, $N \times N$, K é a matriz kernel.

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, $\left\{x_i^{\mathcal{T}}\right\}_{i=1}^N \in \mathcal{T}$ são centralizados em relação $\mathcal{D}$ como,

$$\widetilde{K}^{TESTE} = K^{TESTE} - 1_{N_t} K - K^{TESTE} 1_N + 1_{N_t} K 1_N \tag{60}$$

em que $\widetilde{K}^{TESTE}$ é a matriz kernel para os atributos fora do domínio fonte, K é a matriz kernel, 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 $\widetilde{K}^{\mathcal{T}}$, para indicar a integração dos dados amostrais provenientes do domínio alvo, $\mathcal{T}$.

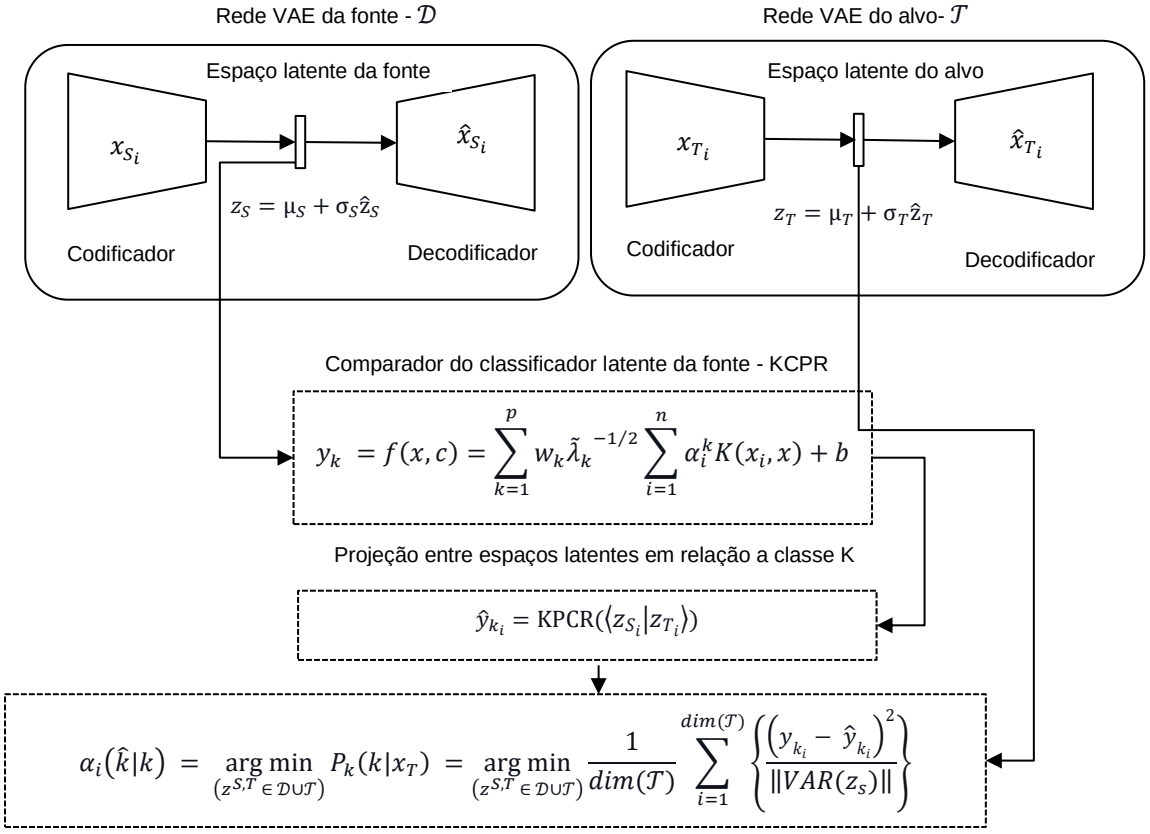
O objetivo do KPCR é extrapolar as classes na transferência de aprendizagem, baseados no espaço latente da rede VAE que proveu um agrupamento semi-supervisionado. Para tanto, as classes foram definidas conforme a notação da codificação *one-hot*, e centralizadas como,

$$y_0 = \left(1_N - \frac{1}{N} 1_N 1_N^T\right) y \tag{61}$$

em que y_0 é o vetor centralizado para as classes. A notação para as k classes foram previamente definidas 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 auto codificadora foram as respostas impulsivas das salas com suas respectivas classes.

FIGURA 15 – FLUXOGRAMA DO ALGORITMO MVPAnP



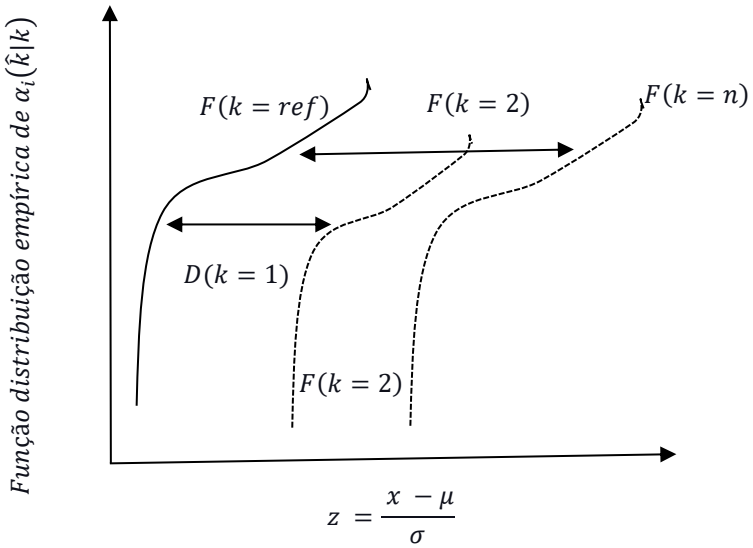
FONTE: O próprio autor (2023).

A solução do problema de minimização mostrada na FIGURA 15 foi realizada pelo algoritmo da evolução diferencial (STORN; PRICE, 1997). A escolha de usar somente reposta impulsiva deve-se do cálculo normatizado do STI que somente a resposta impulsiva. Portanto, delineia-se um classificador não linear baseado na distância das projeções dos espaços latentes, dadas por,

$$\alpha_i(\hat{k}|k) = \underset{(z^{S,T} \in DUT)}{\arg\min}\ P_k(k|x_T) = \underset{(z^{S,T} \in DUT)}{\arg\min}\ \frac{1}{\dim(T)} \sum_{i=1}^{\dim(T)} \left\{ \frac{\left(y_{k_i} - \hat{y}_{k_i}\right)^2}{\left\|VAR(z_s)\right\|} \right\} \tag{62}$$

em que, $\alpha_i(\hat{k}|k)$ é a estimativa das classes no espaço alvo. A FIGURA 16 gráfica a interpretação geométrica do problema de otimização, o qual objetiva minimizar as diferenças entras as distribuições das classes via projeção dos espaços latentes.

FIGURA 16 – INTERPRETAÇÃO GRÁFICA DO TESTE DE SIGNIFICÂNCIA DO MVPAnP



FONTE: O próprio autor (2023).

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 estas separam os algoritmos de agrupamento não supervisionado e os projeta

sobre as classes usando o MMD (LONG et., 2013). Wang, Jing (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 DO MODELO E MEDIÇÕES

Essa seção tem como objetivo descrever a validação experimental do modelo de redes neurais profundas convolucionais aplicada na predição do STI. Além disso, 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.

A validação das predições será realizada com base em dois conjuntos de medições em salas de aula, o que totalizarão 75 salas medidas. Dessa forma, o primeiro conjunto com (n = 60) medições experimentais do TR provenientes do banco de dados de Ko et. al (2017). O segundo com (n = 15) medições do STI e TR em salas de aula da Universidade Federal do Paraná, Campus Centro Politécnico, que não foram apresentadas durante o treinamento das redes neurais.

3.7.1 Medições in situ

As medições do STI serão realizadas seguindo as diretrizes da norma IEC 60268-16 (IEC, 2011), os 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. A FIGURA 17 mostra a disposição da instrumentação dentro de uma sala de aula.

FIGURA 17 – DISPOSIÇÃO EXPERIMENTAL DAS MEDIÇÕES DO STI E TR

FONTE: O próprio autor (2023).

O STI foi medido com os seguintes instrumentos: 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; Lab. 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 é equalizado usando o sinal MLS com o filtro Pink + Blue, conforme recomendado pela IEC 60268-16 (IEC, 2011). Os ganhos no equalizador serão ajustados para o nível de pressão sonora de referência de 60 dB, e com um erro tolerável de ± 1 dB, para as bandas de oitava de 125 Hz e 8 kHz.

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

Para as medições, o simulador da boca artificial será 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.

O tempo de reverberação será 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 dodecaedro B&K 4296 foi posicionada a uma altura de 1,5 m do chão e a mais de 1,2 m das paredes. Cinco medições foram registradas com o receptor (analisador de som B&K 2260) em diferentes posições em cada sala de aula.

3.7.2 Validação

Conforme a seção 3.3.1, adotou-se com a primeira entrada do modelo de redes neurais profunda as respostas impulsivas das salas correspondente ao tempo de reverberação e dessas calculou-se a densidade espectral. A segunda entrada do modelo foi a densidade espectral do ruído de fundo medido nas salas

A validação do modelo de redes neurais profundas para predições do STI foi realizada através do cálculo do Erro Percentual Absoluto Médio, ou em inglês, *Mean Absolute Percentage Error* (MAPE), este erro pondera os valores de referência e os valores obtidos via o modelo de redes neurais profundas, para dados não apresentados para a rede durante o treinamento, utilizando a abordagem de transferência de aprendizagem via MVPAnP. O erro MAPE é dado como,

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \times 100\% \tag{63}$$

em que A_t é o valor de referência, F_t é valor predito pelo modelo de redes neurais profundas. O A_t será calculado como o próprio modelo normatizado segundo a IEC 60268-16 (IEC, 2011) que usa o tempo de reverberação como parâmetro. Tal modelo foi descrito na seção 3.2 aplicando a Eq.19. Após, confeccionou-se a curva de calibração do modelo e calculou-se o coeficiente de correlação de Pearson, dado por,

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - y)^2}} \tag{64}$$

em que ρ é o coeficiente de correlação de Pearson, x_i e y_i são os valores esperados e medidos respectivamente, com i = até n, onde $n = 60$ são o número 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 de resíduos.

Adicionalmente, avaliou os mesmos erros de modelo, perante as medições in situ (n = 15) do STI indireto, seguindo os procedimentos estabelecidos IEC 60268-16 (IEC, 2011) e consolidados na seção 2.2.2.

4 RESULTADOS E DISCUSSÕES

Esta seção apresenta os resultados das medições *in-situ*, dos experimentos computacionais e das análises comparativas frente a literatura corrente. Assim, em primeiro momento, delineia-se a metodologia desenvolvida, seguindo-se de uma descrição dos resultados. Após, discute-se as implicações desses resultados e por conseguinte fornece-se uma interpretação de seu significado usando ferramentas de IA explicada (*eXplainable AI*).

Por fim, compara-se os resultados aqui determinados com trabalhos anteriores relevantes da literatura e destaca-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 PIPELINE DE PROCESSAMENTO DE DADOS

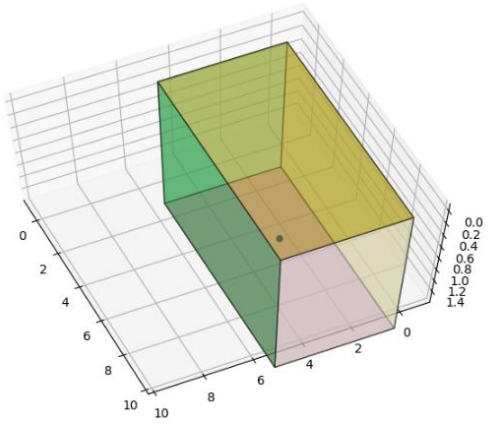
Conforme a seção 3.2, dados sintéticos são os dados gerados artificialmente, em vez de, coletados de observações do mundo real (*in locu*). Assim, dado o contexto da geração de respostas acústicas em salas de aula, dados sintéticos podem ser usados para simular as características sonoras de uma sala de aula, 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.

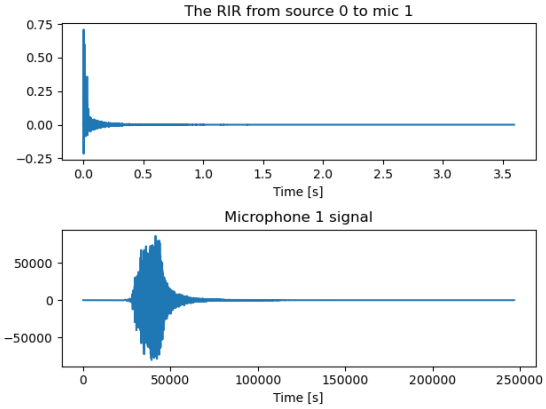
Assim, de acordo com a seção 3.2.1, o método de fonte de imagem foi usado para gerar 10.000 salas virtuais com propriedades acústicas únicas. A FIGURA 18 ilustra um exemplo do modelo acústico e a resposta ao impulso de sala (RIR) correspondente, capturada na posição do receptor.

FIGURA 18 – MODELO VIRTUAL DAS SALAS DE AULA RENTAGULARES

a) Sala virtual



b) Pressão Sonora na posição do receptor



LEGENDA: Sala virtual gerada pelo Método das Imagens (ver seção 3.2.1) (b) Exemplo de captura RIR e uma posição aleatória dentro da sala.

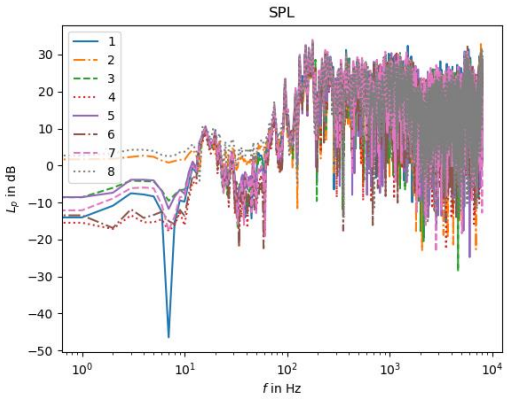
FONTE: O próprio autor (2023).

A FIGURA 18 apresenta a distribuição em banda larga do nível de ruído de fundo extraído de uma amostra de áudio do banco de dados de Ko et al. (2017) que foi convolado digitalmente na sala virtual. 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. 18 e 19. Esta figura fornece uma representação visual da acústica da sala virtual e como as ondas sonoras são refletidas/absorvidas dentro dela.

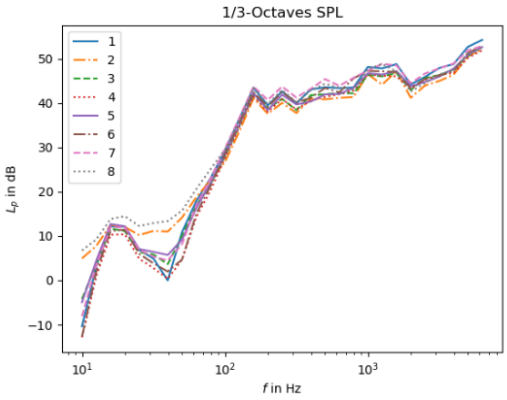
Adicionalmente, a FIGURA 19 fornece uma representação gráfica do nível de pressão sonora da RIR nas amostras de áudio de áudio (brutos e sintéticos), assim como o resultado da aplicação do filtro de 1/3 de oitava. Essa figura demonstra a diferença de como os sinais sintéticos respondem ao filtro de 1/3 de oitiva.

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

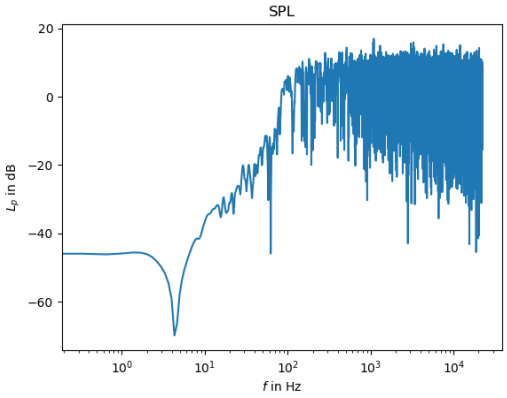
a) Nível de Pressão Sonora (medição/in situ)



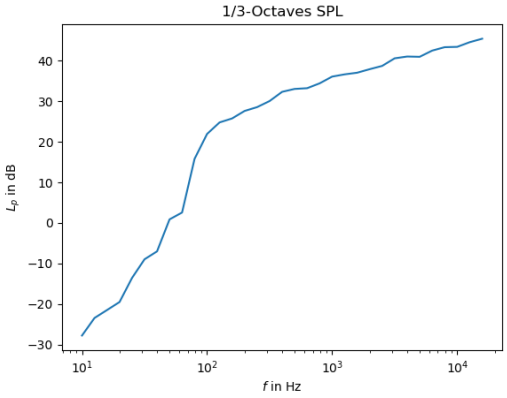
b) Nível de Pressão Sonora em 1/3 de oitava



c) Nível de Pressão Sonora (dado sintético)



d) Nível de Pressão Sonora em 1/3 de oitava



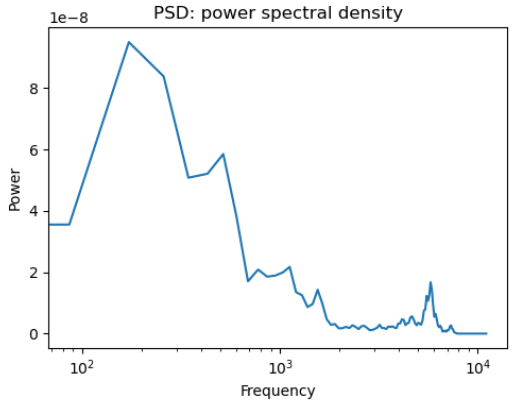
LEGENDA: Comparação das medições da RIR in situ versus dados sintéticos obtidos de (a) Sinal acústico de excitação da banda larga extraído do banco de dados de ruído ambiental de Ko et al. (2017), (b) Sinal acústico filtro de banda de 1/3 de oitava.

FONTE: O próprio autor (2023).

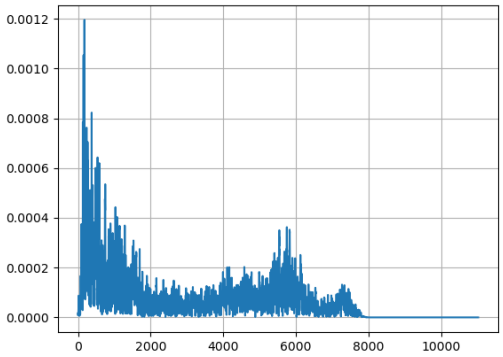
De posse das curvas RIR e do ruído de fundo de banda larga, conforme mostrado na FIGURA 18 e o correspondente valor de STI, estes foram submetidos como as entradas do modelo de aprendizagem profunda (DNN) para a realização do treinamento tanto para os modelos, com ou sem transferência de aprendizagem. Portanto, a FIGURA 20 exemplifica um par de entrada de uma amostra do PSD do RIR e do PSD do sinal de ruído de fundo (BGN) do banco de dados de fonte sintética (conjunto de origem) e o conjunto de dados de destino, respectivamente. Assim como na FIGURA 19 a disposição de energia do sinal sintético apresentou-se de forma mais uniforme do que os dados de medidas in situ.

FIGURA 20 – COMPARAÇÃO DAS CURVA RIR IN SITU VERSUS DADOS SINTÉTICOS

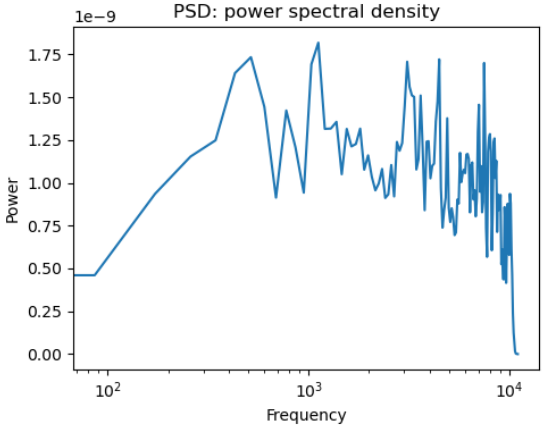
a) Densidade espectral (medição/in situ) - RIR



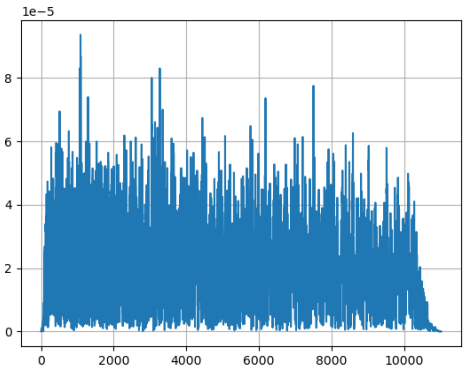
b) Densidade espectral (medição/in situ) - BGN



d) Densidade espectral (dado sintético) - RIR



d) Densidade espectral (dado sintético) - BGN



LEGENDA: Exemplo de PSD e RIR (a) Exemplo de PSD do RIR simulado (b) Exemplo de PSD do sinal de ruído de fundo.

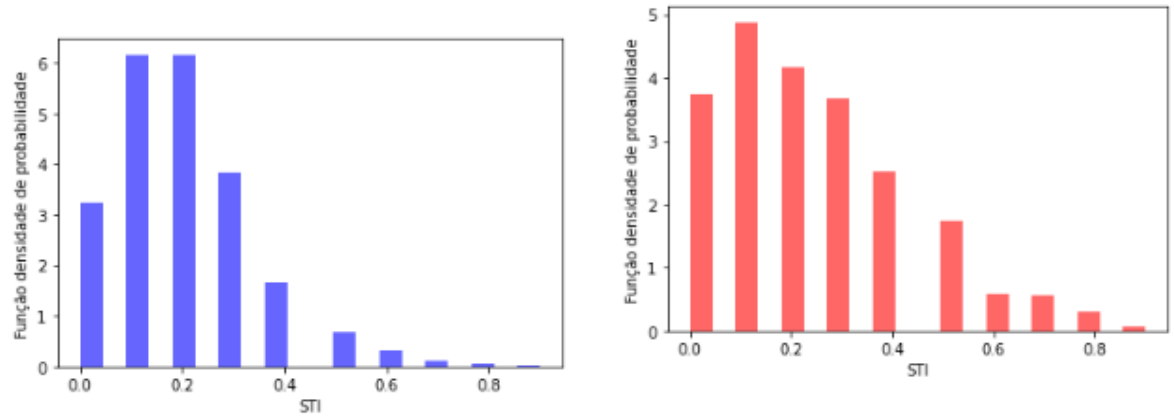
FONTE: O próprio autor (2023).

A FIGURA 21 exibe o histograma dos valores de STI para as 10.000 amostras e 10 classes do STI. Esses resultados se alinham com os resultados de que os valores de STI estão usualmente concentrados na faixa de 0,4 a 0,6. Esta figura fornece uma representação visual da distribuição dos valores de STI dentro do conjunto de dados, permitindo uma comparação direta com o intervalo esperado conforme Bradley (2009).

FIGURA 21 – COMPARAÇÃO DAS DISTRIBUIÇÇÕES DO STI MEDIDO VERSUS SIMULADO

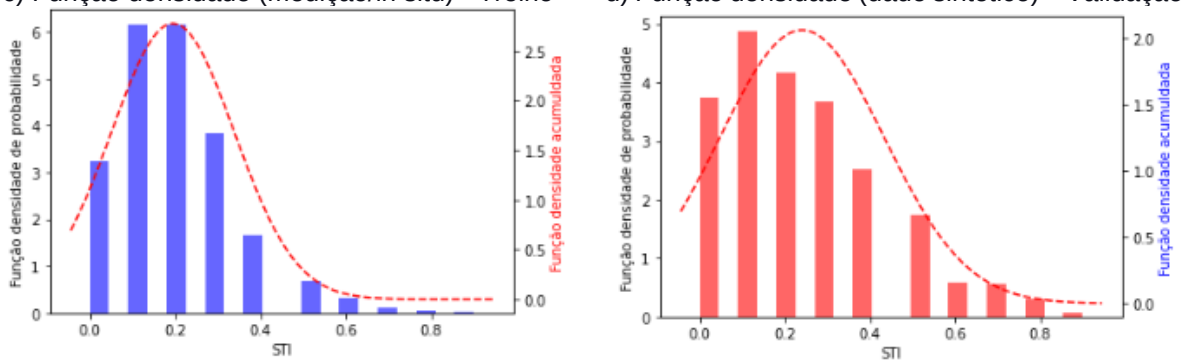
a) Histograma (medição/in situ) - Treino

b) Histograma (dado sintético) – Validação



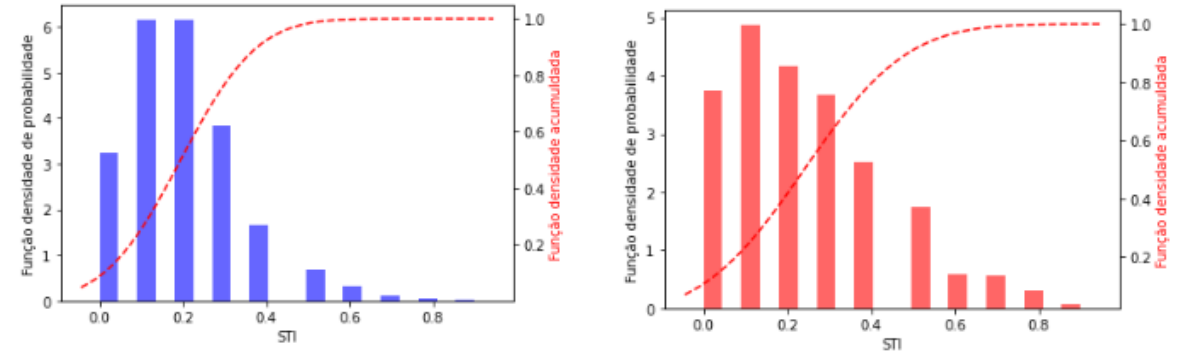
c) Função densidade (medição/in situ) - Treino

d) Função densidade (dado sintético) – Validação



e) Função acumulada (medição/in situ) - Treino

f) Função densidade (dado sintético) - Validação



LEGENDA: Histograma da amostra de treinamento para STI. Figuras do lado esquerdo pertencem ao domínio de origem, enquanto, os gráficos da direita pertencem domínio de destino. (a) Histograma, (c) Função Densidade de probabilidade (FDP) (e) Função de Distribuição Acumulada (FDA)

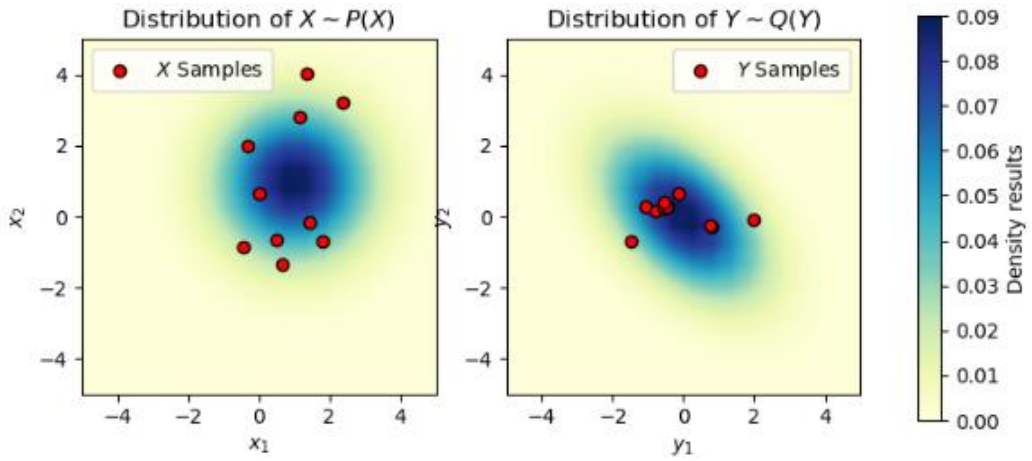
FONTE: O próprio autor (2023).

De acordo com a FIGURA 21, é possível avaliar visualmente uma similaridade estatística, na perspectiva Gaussiana, entre os histogramas STI dos domínios alvo

(média = 2,4, sigma = 1,93) e fonte (média = 1,94, sigma = 1,43). Para validar quantitativamente a similaridade acima declarada calculou-se a métrica MMD. O valor da medida MMD foi de 0,53, conforme mostrado na FIGURA 22.

Como resultado, uma vez que o MMD é considerado uma medida prática e confiável de diferença distributiva por ser não paramétrico, ou seja, não assume nenhuma forma funcional para as distribuições subjacentes. Isso o torna robusto para diferentes tipos de diferenças distribucionais e permite que seja aplicado a uma ampla gama de conjuntos de dados como o STI. Ademais, como validação semiempírica, foi possível aplicar a transferência de aprendizagem profunda, 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).

FIGURA 22 – MÉTRICA MMD PARA DOMÍNIO DE ORIGEM E DESTINO

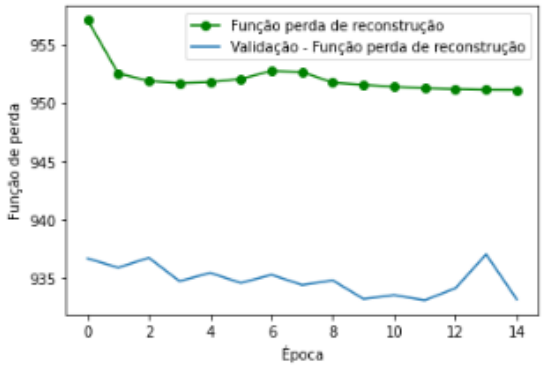


FONTE: O próprio autor (2023).

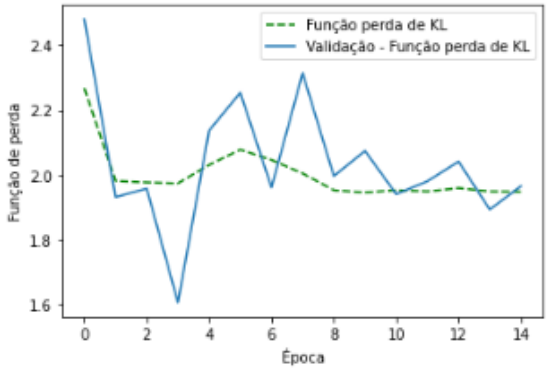
Depois que os conjuntos de dados de origem e destino foram gerados, procedeu-se ao treinamento de uma rede neural profunda usando um modelo de auto codificador variacional (VAE) para obter o posterior aprendizado sobre o espaço latente (*embedding learning)* para a aplicação Transferência de Aprendizagem (TF). A FIGURA 23 exibe as métricas da função de perda global (soma dos componentes) conforme a Eq. 48.

FIGURA 23 –MÉTRICA DE RECONSTRUÇÃO DE PERDA DURANTE O TREINAMENTO

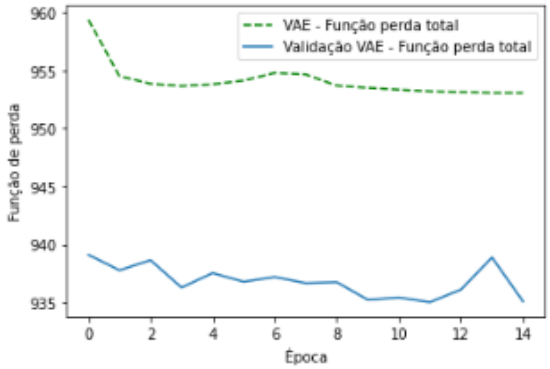
a) Função de perda – reconstrução



b) Função de perda – KL perda



c) Função de perda total



LEGENDA: Perda de VAE no conjunto de dados de treinamento para reconstrução VAE-CNN-DNN: (a) Época de treino versus Função perda – Perda de reconstrução e perda e b) Função perda métrica de divergência de Kullback-Leibler, (c) Função de perda soma ponderada.

FONTE: O próprio autor (2023).

Portanto, 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 +

perda 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 aprendizado por transferência, 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 a seção 4.2.

4.2 OTIMIZAÇÃO DOS HIPERPARÂMETROS DA REDE AUTO CODIFICADORA VARIACIONAL

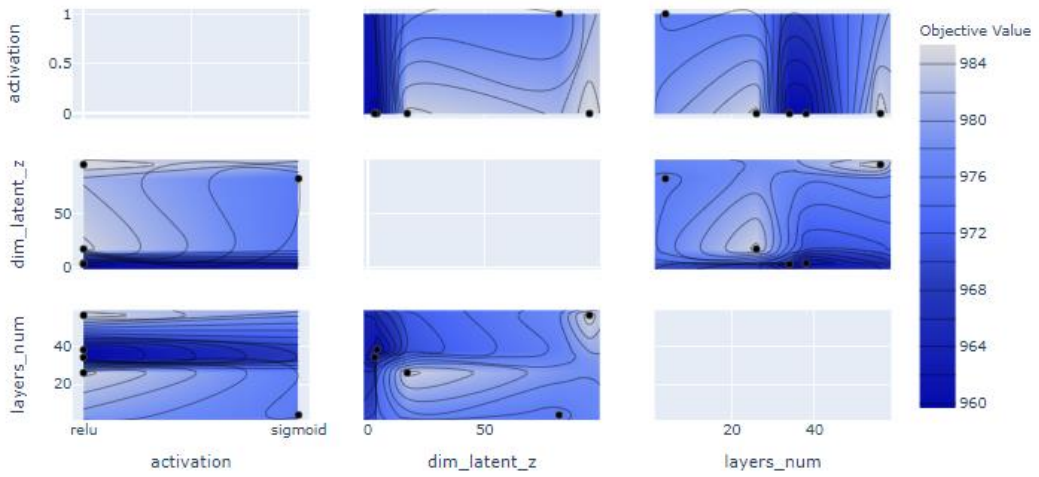
Uma vez que neste trabalho, explorou-se a influência de diferentes parâmetros na precisão do codificador de saída (camada de saída da rede autocoficadora). Os resultados indicam que os fatores mais cruciais que afetam os valores da função custo são as dimensões do espaço latente (z) e o número de neurônios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

Para avaliar completamente o efeito de diferentes hiperparâmetros na precisão do modelo, examinou-se as combinações aleatória da malha de busca desses parâmetros e calculou-se os valores resultantes da função de custo. Os resultados são apresentados na TABELA 4 e visualizados na FIGURA 24 como curvas de contorno.

FIGURA 24 – CURVAS DE CONTORNO DA OTIMIZAÇÃO DOS HIPERPARÂMETROS DO VAE



LEGENDA: Gráfico de contorno para visualizar a otimização de hiper parâmetros para o modelo de aprendizagem profunda convolucional auto codificador variacional.

FONTE: O próprio autor (2023).

Ao inspecionar o resultado da FIGURA 24, observa-se que aumentar o número de camadas ocultas e neurônios na primeira camada leva a uma melhoria na métrica da função de reconstrução. Ao passo que a alteração das funções de ativação não alterou o valor da função custo, indicando desse modo, que aumentar a complexidade via número de neurônios, ajuda a capturar padrões mais sutis nos dados e, por fim, melhorar o desempenho.

Ao realizar a validação cruzada em cinco execuções (CV = 5), verificou-se que a métrica de reconstrução permaneceu estável com um baixo desvio padrão. De tal modo, conforme os parâmetros de otimização foram variados, mostrado na TABELA 3 na Seção 3. Esses resultados destacam a importância de selecionar o hiperparâmetros apropriados para alcançar o desempenho ideal para a reconstrução do modelo variacional conforme mostra a TABELA 4.

TABELA 4 – HPO DO VAE

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

* (a) Nominal função de ativação é discreta; (b) número inteiro, qualquer valor no intervalo dado.

FONTE: O autor (2023).

A TABELA 4 fornece evidências de que a otimização dos hiperparâmetros aumento a precisão do modelo VAE para o problema de reconstrução do codificador-decodificador, atingindo até 95% no conjunto de dados de treinamento. Além disso, esta tabela revela as combinações ótimas de neurônios de entrada e dimensão latente z que foram determinadas por meio do processo de otimização de hiperparâmetros.

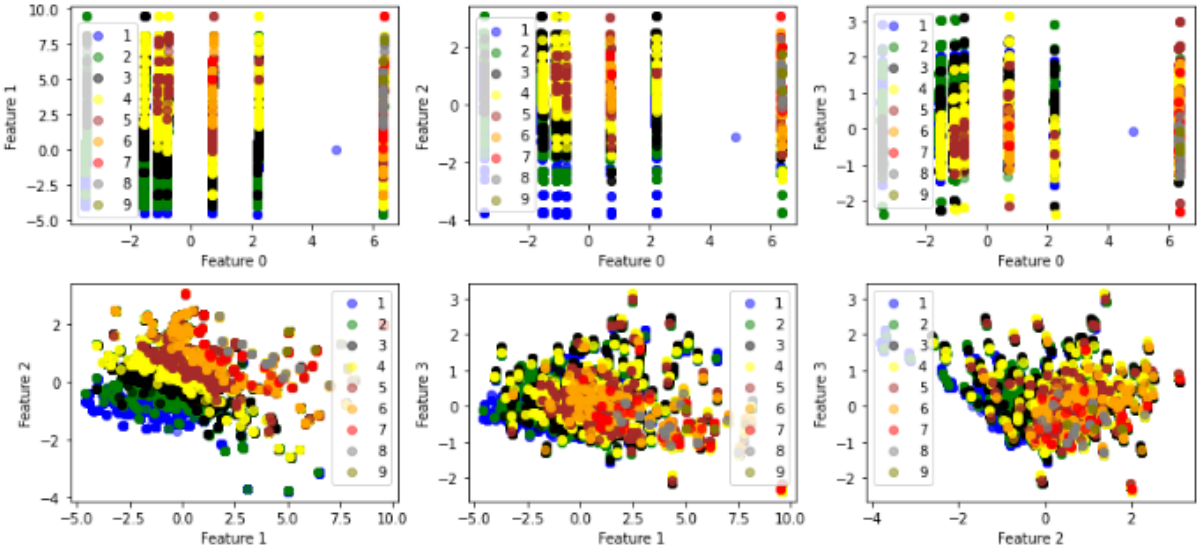
Adicionalmente, verificou-se que o número de dimensões é um parâmetro crítico para o modelo VAE, pois determina o nível de compressão aplicado ao conjunto de dados. Além disso, os conceitos de IA explicável discutidos na Seção 5.2 foram utilizados para avaliar a interpretabilidade do espaço latente ótimo calculado. 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

Depois de treinar o modelo VAE e selecionar a configuração ideal por meio da otimização de hiperparâmetros, foi identificado a distribuição estatística para o conjunto treinamento e procede-se para a aplicação do aprendizado de transferência profunda para determinar o estado representacional de cada conjunto de dados de origem/destino. Em seguida, aplica-se o cálculo de perda (MMD) para avaliar o desempenho do modelo nesses conjuntos de dados de acordo com a FIGURA 25.

O estado representacional, é uma proxy da distribuição da projeção do conjunto de origem sobre um espaço bidimensional. Para tanto, tem-se na FIGURA 25 a representação da projeção do PCA, é possível avaliar que o PCA não capturou o suficiente da variância dos dados, pois os clusters gerados pelo PCA estão sobrepostos e mal separados, dificultando o discernimento de quaisquer padrões ou tendências claras nos dados.

FIGURA 25 – DISPERSÃO DA PCA SOBRE PSD E BGN EM 4 COMPONENTES PRINCIPAIS



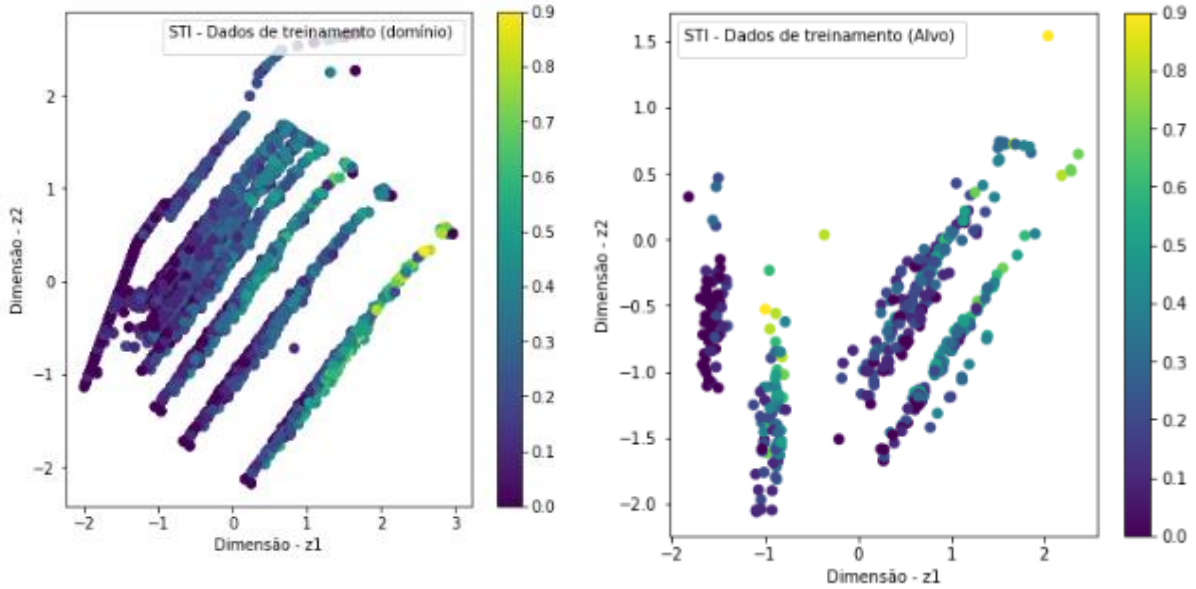
LEGENDA: Projeção bidimensional do espaço das características (componentes principais). Verifica-se que os dois principais componentes não foram capazes de abstrair uma representação definida de separabilidade dos agrupamentos.

FONTE: O próprio autor (2023).

Nessa mesma linha de análise, porém abarcando a representação embutida após o treinamento da VAE. A FIGURA 26 apresenta os valores das classes atribuídas ao espaço latente, com uma projeção nas duas primeiras dimensões usando o VAE. Esta figura fornece uma representação visual da relação entre as classes e sua representação no espaço latente, permitindo uma compreensão visual da dispersão após modelo gerar um novo espaço das características (*feature space*) para o conjunto alvo, baseado somente no conjunto alvo.

FIGURA 26 – PROJEÇÃO DE RECURSO NÃO PROCESSADO KPCA VERSUS PCA

a) Espaço latente do conjunto de domínio b) Espaço latente do conjunto de validação



LEGENDA: Gráfico da projeção bidimensional do espaço latente (componentes principais não lineares). Verifica-se que os dois principais componentes foram capazes de abstrair uma representação definida de separabilidade dos agrupamentos para as classes dos valores correspondentes ao STI.

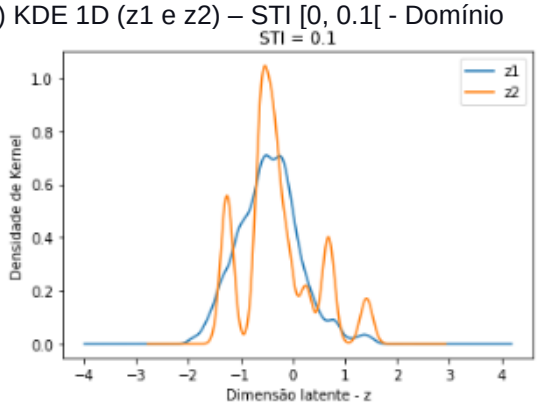
FONTE: O próprio autor (2023).

Como mencionado na seção 3.5, dois conjuntos de dados foram usados neste estudo: (i) o conjunto de dados de treinamento e o (ii) conjunto de dados de teste. O conjunto de dados de teste não foi usado durante o treinamento da rede neural profunda DNN-CNN-VAE. Como resultado, a distribuição de classes foi uniforme para ambos os conjuntos de dados. A arquitetura do autoencoder foi então utilizada para construir uma representação simplificada das classes no espaço latente, que foi projetada em um espaço bidimensional.

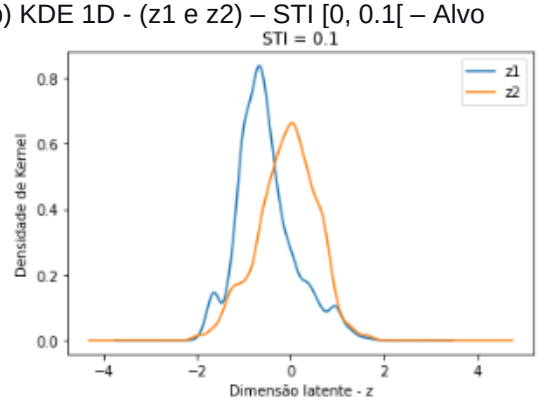
A FIGURA 27(a) mostra um exemplo da classe (k = 1) para o conjunto de dados de treinamento, enquanto a FIGURA 27(b) mostra a mesma classe, mas nos dados não vistos no conjunto de dados de teste. Essas figuras fornecem visualizações da capacidade do modelo de representar as classes no espaço latente e de generalizar para dados apresentados durante o treinamento.

FIGURA 27 – COORDENADAS LATENTES VERSUS DENSIDADE DE NÚCLEO VIA KDE

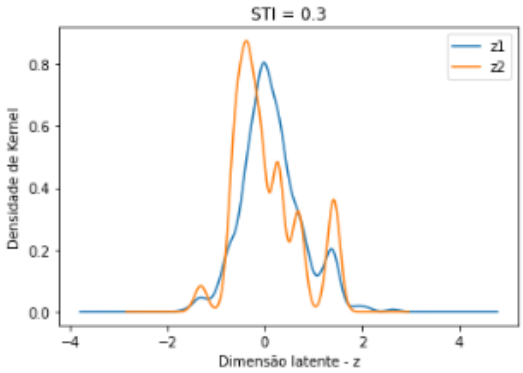
a) KDE 1D (z1 e z2) – STI [0, 0.1[- Domínio



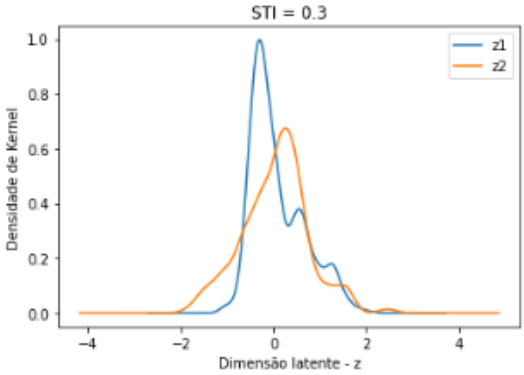
b) KDE 1D - (z1 e z2) – STI [0, 0.1[– Alvo



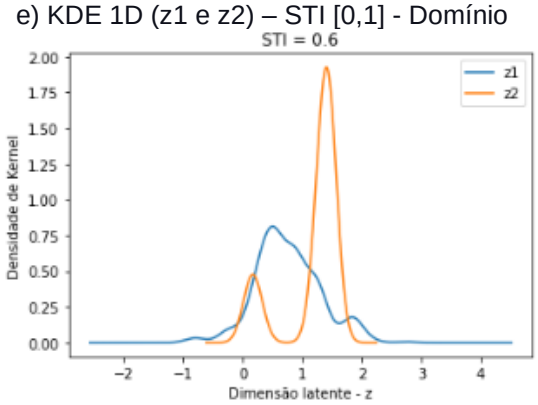
e) KDE 1D (z1 e z2) – STI [0,1] - Domínio



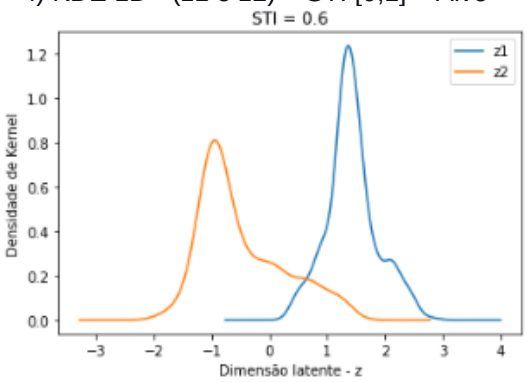
f) KDE 1D - (z1 e z2) – STI [0,1] – Alvo



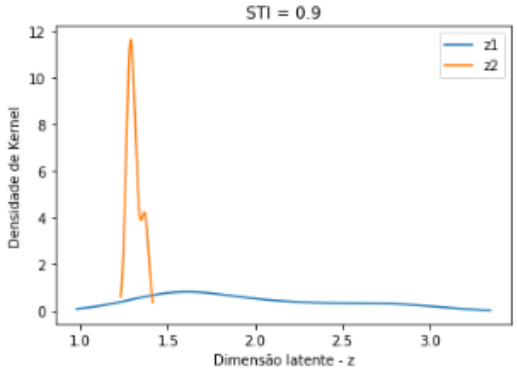
e) KDE 1D (z1 e z2) – STI [0,1] - Domínio



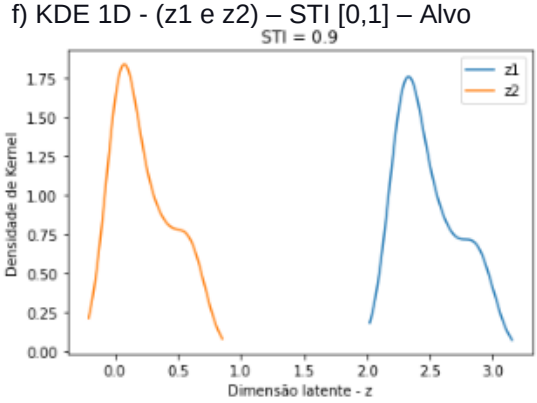
f) KDE 1D - (z1 e z2) – STI [0,1] – Alvo



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



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



FONTE: O próprio autor (2023).

Comparando as coordenadas latentes z1 e z2 do MMD, as FIGURAS 27(a) e 27(b) revelam um nível proeminente de concordância. A correlação linear quadrada para a coordenada z1 é 0,95 e para a coordenada z2 é 0,95. Como resultado, usando a equação 9, a classe predita foi atribuída corretamente porque a distância métrica MMD estava próxima de zero, indicando altos valores de correlações lineares.

A TABELA 5 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 5 – TESTE DE KOLMOGOROV-SMIRNOV PARA TESTE DE DIFERENCIABILIDADE

Camada	Nome da camada	Função de Ativação	Hiper parâmetro
7	Global average pooling	#	#
8	Dropout	#	Razão de corte: 0,1; 0,25 e 0,50
10	Camada MLP	ReLu	Número de neurônios: 10; 100 e 200
11	Saída do FC	Sofmax	20

FONTE: O autor (2023).

Conforme a TABELA 5, todas as estatísticas D destacaram grande concordância. O mesmo resultado foi observado para as demais classes. Esses resultados demonstram a eficácia da métrica MMD em prever com precisão as classes com base nas coordenadas latentes.

4.3 IA EXPLICÁVEL APLICADA AO ESPAÇO LATENTE

Comparando as coordenadas latentes z1 e z2 do MMD, as FIGURA 27(a) e 27(b) revelam um nível elevado de concordância com base no MMD (FIGURA 22) e nos valores do teste KS, ver TABELA 5. No entanto, para obter uma compreensão mais profunda do que está acontecendo dentro do modelo, é essencial fornecer explicabilidade ao espaço de aprendizagem representacional.

Neste trabalho foram avaliadas duas abordagens que foram aplicadas para atingir esse objetivo: (i) primeira abordagem foi a otimização de hiperparâmetros e a (ii) segunda abordagem foi a análise de sensibilidade da distribuição do espaço latente, conforme mostrado na TABELA 6 Essas abordagens permitem uma

compreensão mais completa do comportamento do modelo e melhoram sua interpretabilidade.

TABELA 6 – ANÁLISE DE SENSIBILIDADE SORE OS HIPERPARÂMETROS DE VAE

Parâmetro	Função custo	Tempo de treinamento
Dimensão espaço latente	0,60	0,49
Número de neurônios nas camadas MLP de ocultas	0,26	0,40
Função de ativação	0,14	0,10

* variância explicada sobre o valor da função custo

FONTE: O autor (2023).

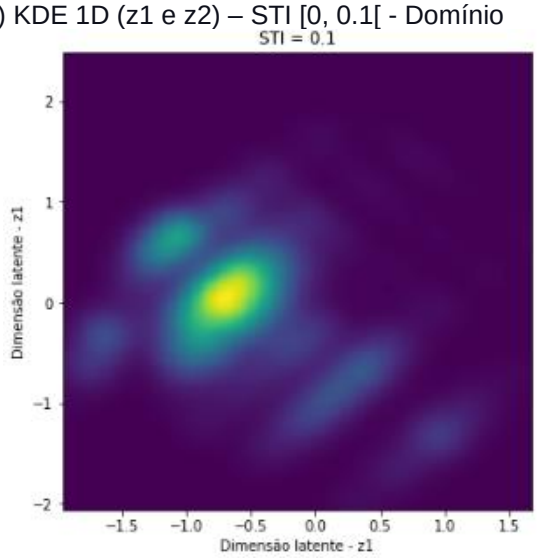
Baseando-se na TABELA 6, a variância explicável de 37% indica que a dimensão do espaço latente foi um dos principais fatores que afetam a precisão do modelo. Em contraste, o número de camadas ocultas teve pouco ou nenhum impacto na precisão. Esse resultado é consistente com os paradigmas de aprendizado de kernel e aprendizado de máquina incorporados na literatura, que enfatiza a importância das dimensões do espaço latente que geram uma variedade não linear, em vez das topologias e estruturas de modelos específicos.

Essas descobertas fornecem informações valiosas sobre os fatores que influenciam a precisão do modelo e dão suporte ao uso da abordagem de desvio nesse contexto. Uma vez que a minimização da máxima discrepância média (MMD) é um método usado no aprendizado de transferência para alinhar as distribuições estatísticas dos domínios de destino e de origem. Ele faz isso minimizando a distância entre as duas distribuições, conforme medido pela estatística MMD. 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 28.

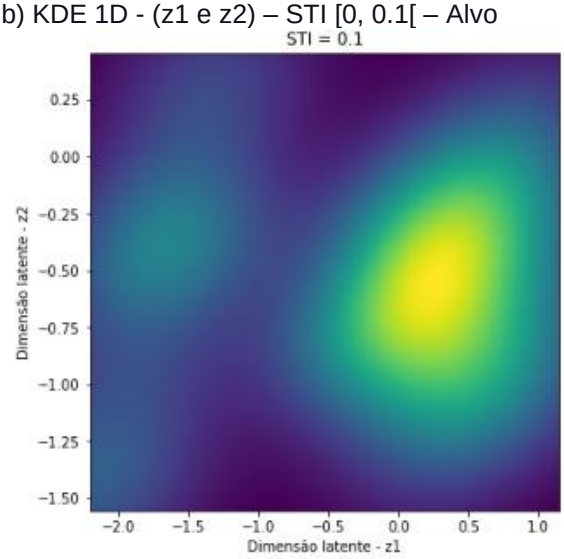
Um gráfico do KDE bidimensional é uma forma não paramétrica de estimar a função de densidade de probabilidade de uma variável aleatória e pode ser usado para visualizar a forma de uma distribuição. O uso de um gráfico do KDE pode fornecer uma representação visual do alinhamento das distribuições e da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

FIGURA 28 – GRÁFICO DE COORDENADAS LATENTES DA DISTRIBUIÇÃO DO KDE

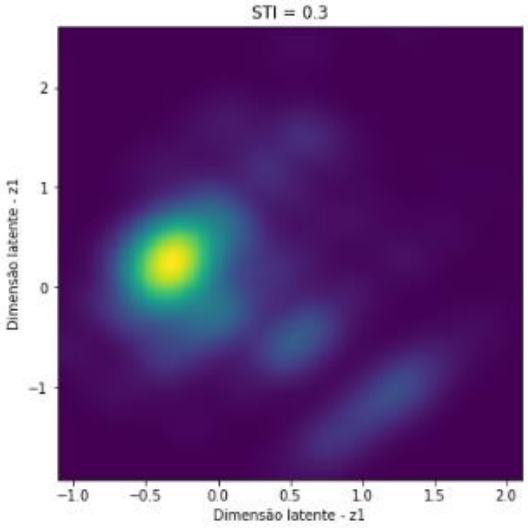
a) KDE 1D (z1 e z2) – STI [0, 0.1[- Domínio



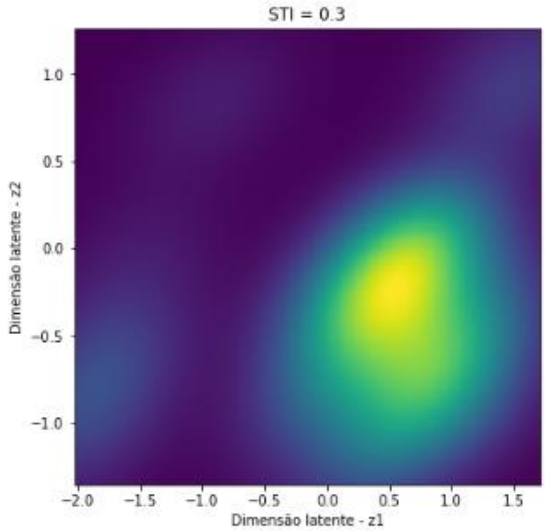
b) KDE 1D - (z1 e z2) – STI [0, 0.1[– Alvo



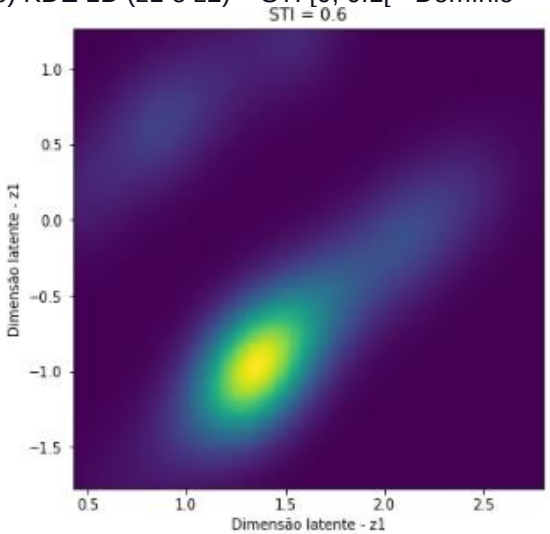
c) KDE 1D (z1 e z2) – STI [0, 0.1[- Domínio



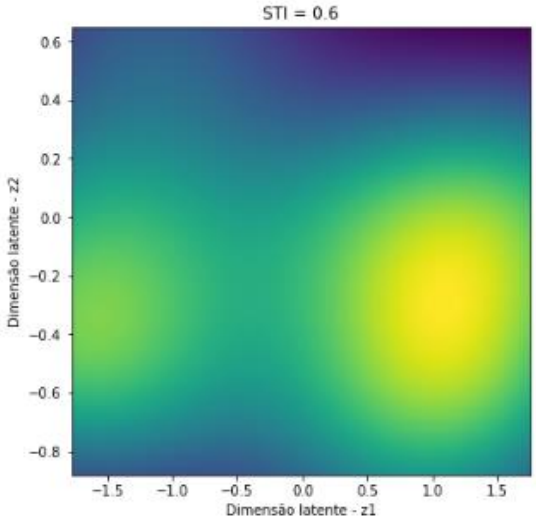
d) KDE 1D - (z1 e z2) – STI [0, 0.1[– Alvo



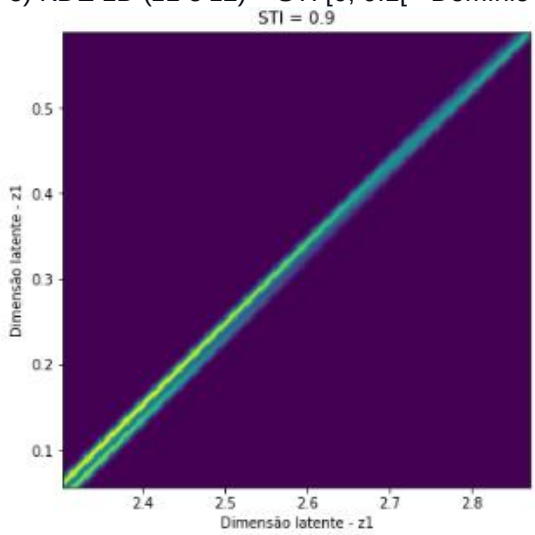
e) KDE 1D (z1 e z2) – STI [0, 0.1[- Domínio



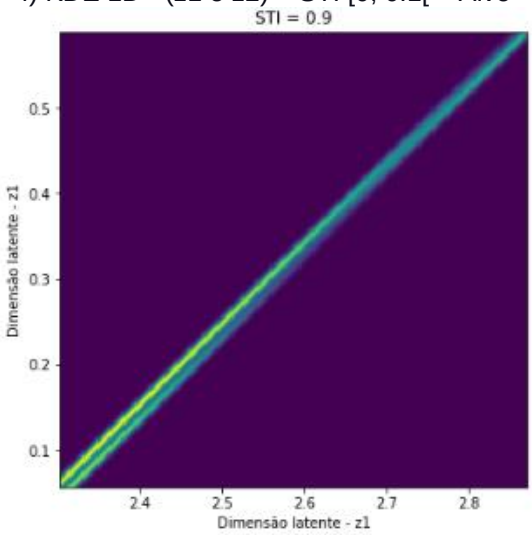
f) KDE 1D - (z1 e z2) – STI [0, 0.1[– Alvo



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



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

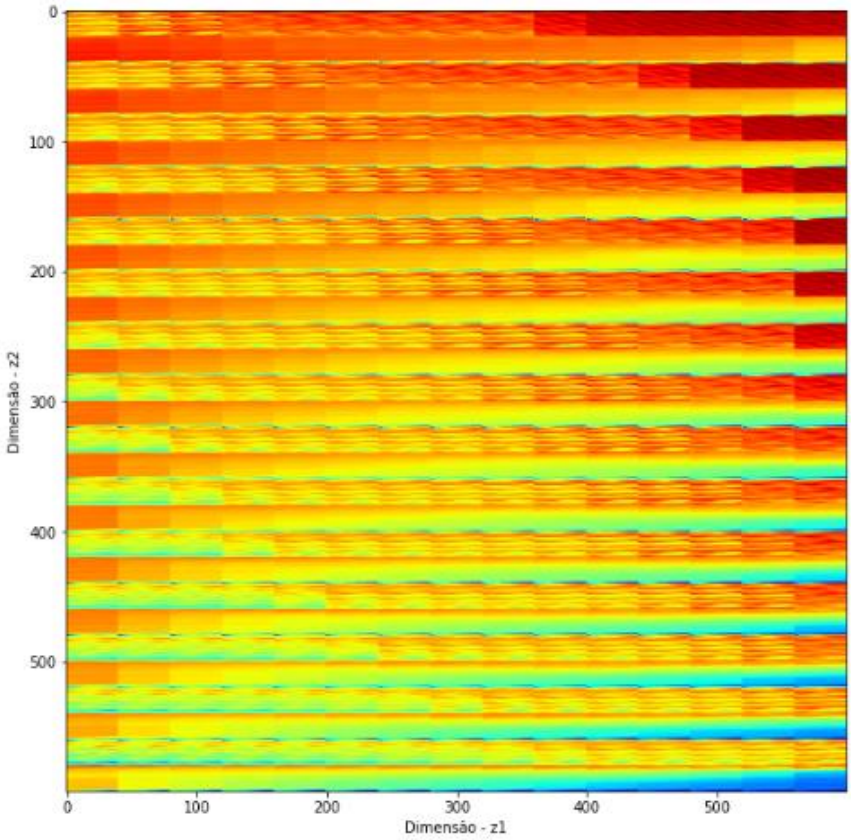


FONTE: O próprio autor (2023).

Consequentemente, baseando no gráfico KDE da FIGURA 29, comparou-se a eficácia do estado representacional usando o modelo Depp-VAE. De acordo com Fulano (2022), no contexto do aprendizado por transferência, um gráfico do KDE pode ser usado para comparar as distribuições dos domínios de origem e destino antes e depois da minimização do MMD.

Se as distribuições estiverem bem alinhadas, os gráficos do KDE dos domínios de origem e destino devem ser semelhantes. Por outro lado, se as distribuições não estiverem bem alinhadas, os gráficos do KDE serão diferentes, indicando uma discrepância entre as duas distribuições. Além disso, a FIGURA 29 destaca o conhecimento de auto-organização topológica obtido das coordenadas do espaço latente.

FIGURA 29 – ESPAÇO LATENTE RELAÇÃO ENTRE O RUÍDO DE FUNDO E REVERBERAÇÃO



FONTE: O próprio autor (2023).

Conforme a FIGURA 29, observa-se que as variações do espaço de entrada 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 foram atribuídos ao lado esquerdo do gráfico, enquanto os valores mais baixos foram atribuídos ao lado direito do gráfico.

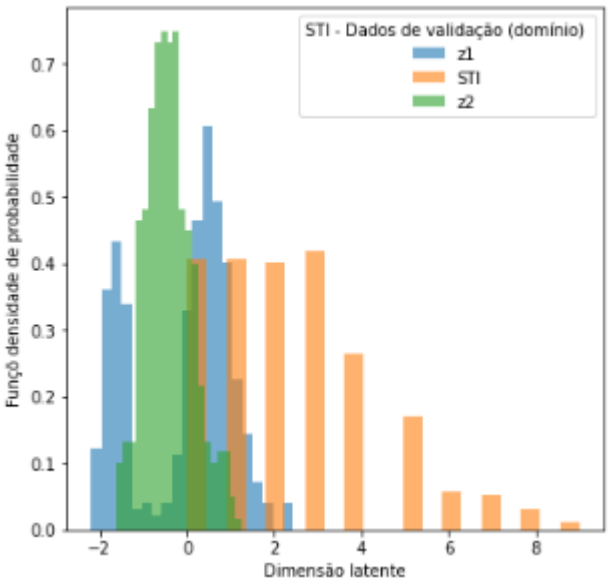
Esses resultados são consistentes com a literatura atual e demonstram que o espaço latente pode refletir as distribuições de classes mostradas na FIGURA 26. Além disso, fica evidente a distribuição desbalanceada de classes entre os conjuntos de dados, com mais classes sendo atribuídas ao intervalo entre zero e um. Este resultado concorda com os achados de Bistafa (2022).

4.4 IMPLEMENTAÇÃO DE TRANSFERÊNCIA DE APRENDIZAGEM

Conforme descrito na Seção 3, nosso método proposto foi comparado com métodos bem conhecidos usados na literatura. Nossos resultados foram normalizados e demonstramos as melhorias na precisão da pontuação do MSE e no incremento relativo do erro MRSE.

As funções de kernel usando filtros gaussianos também foram otimizadas e, para comparações não enviesadas foram empregados os mesmos hiperparâmetros descritos na TABELA 4. A FIGURA 30 mostra o gráfico de distribuição final.

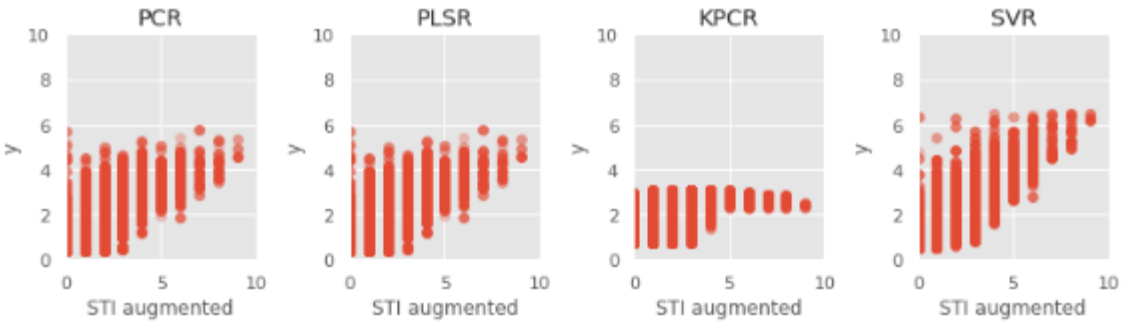
FIGURA 30 – HISTOGRAMA – COORDENADAS DO ESPAÇO LATENTE



FONTE: O próprio autor (2023).

Portanto, com base no teste ANOVA usando o algoritmo MVPanP como hipótese nula, com ($p < 0,01$) e 95% de certeza, o modelo MVPanP usando KPCR melhorou significativamente a precisão do modelo no conjunto de dados de teste. Como comparação, a FIGURA 31 avalia os métodos de benchmarking 4 métodos para servir como a função de mapeamento de acordo com a Eq. 1. No ANEXO 2, encontra-se as análises estatísticas dos ajustes aplicados.

FIGURA 31 – MODELO KCPR APLICADO NO ESPAÇO LATENTE



LEGENDA: Gráfico do ajuste e comparação de modelo de regresso multivariados na estimativa do STI. As dimensões de teste foram de 128. Além disso análises de correlação verificarão alta colinearidade nos modelos.

FONTE: O próprio autor (2023).

Dado o espaço de probabilidade de distribuição do espaço latente, foi possível construir uma linha de regressão usando KPCR para o conjunto de dados de treinamento, o conjunto de dados de validação e o conjunto de dados de destino. Os resultados mostraram uma melhora no problema de classificação usando nossa abordagem proposta.

Os principais resultados estão destacados nas FIGURA 31, onde compara-se o desempenho do nosso método com e sem transferência de aprendizagem. A melhoria no método KPCR foi dez vezes maior. Esses achados demonstram a eficácia dessa abordagem proposta para melhorar a precisão da classificação.

Para avaliar o MVPAnP em relação aos métodos de benchmarking, comparamos as configurações de classificação, a CNN 1D no domínio de origem e comparamos o resultado no domínio de destino no modo de inferência. Além disso, foi usado o vanilla 2D CONV NET usado o espectro MFFC como valores de entrada. Buscando fazer uma comparação justa para os modelos de classificação, o valor Max SoftMax nos valores de saída foi considerado o valor STI previsto, conforme a Fig. 18.

A TABELA 5 mostra uma comparação entre diferentes métodos aplicados à abordagem de aprendizado de transferência profunda usando nosso algoritmo proposto, conforme descrito na Seção 5. Os resultados mostram melhorias significativas em todos os modelos ao considerar fatores como sensibilidade de

entrada, dimensão do espaço latente, distribuições estatísticas no espaço latente e uma série de hiperparâmetros a serem otimizados.

Além disso, a qualidade do modelo foi avaliada usando métricas comparativas como o MSE, tempo de execução. A validação cruzada com CV = 10 foi usada para calcular o erro relativo. Esses resultados demonstram a eficácia do nosso algoritmo proposto em melhorar o desempenho dos modelos de aprendizado de transferência profunda.

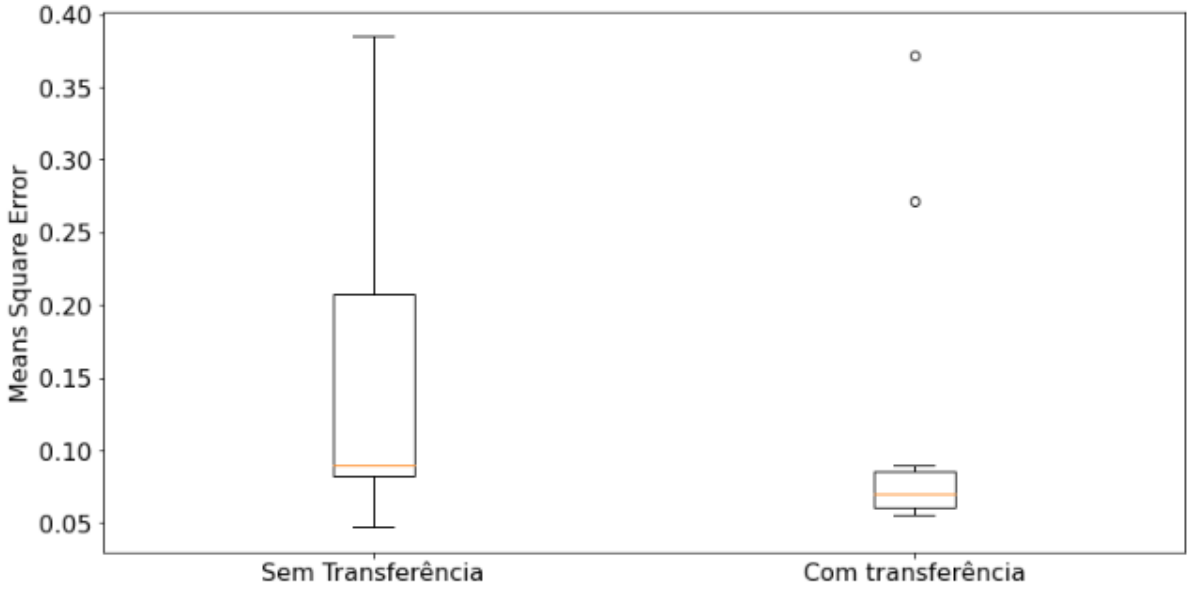
TABELA 7 – VALIDAÇÃO DE MÉTODOS DE TF TRANSFERÊNCIA VIA HIPER PARÂMETROS

Rodadada	**Duração (s)**	**Função de Ativação**	**Dropout**	**Filtros Conv**	**Dimensao do Kernel**	**Strides**	**Loss (MSE)**
1	37.10	sigmoid	0.3	64	5	2	Sem TF
2	35.016	sigmoid	0.2	32	5	1	Sem TF
3	35.591	sigmoid	0.3	64	5	1	Sem TF
5	36.751	swish	0.3	64	5	2	Sem TF
6	36.319	swish	0.3	64	3	1	Sem TF
7	36.059	relu	0.2	64	3	2	Sem TF
8	36.459	swish	0.1	64	5	1	Sem TF
9	36.671	swish	0.3	32	3	1	Sem TF
10	35.360	relu	0.1	64	5	2	Sem TF
1	35.320	sigmoid	0.2	32	5	1	Sem TF
2	47.83	relu	0.1	64	3	2	Com TF
3	47.786	relu	0.3	64	5	2	Com TF
5	47.496	sigmoid	0.2	64	5	2	Com TF
6	47.115	relu	0.2	32	3	2	Com TF
7	47.248	swish	0.1	32	3	2	Com TF
8	47.039	sigmoid	0.3	32	3	1	Com TF
9	46.505	sigmoid	0.2	32	5	1	Com TF
10	46.624	relu	0.2	64	5	2	Com TF

FONTE: O autor (2023).

O método proposto neste artigo 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, que obriga cada indivíduo posterior a estar próximo do anterior. Assim, a FIGURA 32 demonstra a comparação do entre o treinamento com e sem transferência de aprendizagem. Nesta mostra-se que o algoritmo MVPAnP apresentou um melhor desempenho em conjuntos de dados de alvo e oferecemos uma interpretação do termo de regularização em seu objetivo.

FIGURA 32 – Comparação entre os métodos de Transferência de Aprendizagem



FONTE: O próprio autor (2023).

Finalmente, é possível mostrar que nesse trabalho apresenta um método para aprendizado de transferência para recursos de observações não rotuladas, chamado algoritmo MVPanP. Este método é baseado no modelo VAE e impõe que a distribuição posterior agregada esteja próxima a distribuição dos dados do conjunto alvo.

Dessa forma, o método MVPnP atuou como uma proxy de correspondência de classes, 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.

5 CONSIDERAÇÕES FINAIS

Este estudo aplicou o modelo VAE (Variational Autoencoder) 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 via o método proposto atingiu seus objetivos.

5.1 CONCLUSÕES

O modelo VAE foi treinado e otimizado com hiperparâmetros, e o aprendizado de transferência foi aplicado para determinar o estado representacional de cada conjunto de dados de origem/destino. O desempenho do modelo foi avaliado com cálculos de função de perda. A projeção PCA dos dados mostrou que o PCA não capturou suficiente da variância dos dados, enquanto a projeção VAE das classes 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, o primeiro utilizado usado 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 para 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, desenvolvimento de um novo método de transferência de aprendizagem profunda. Esse método, que será denominado de transferência de aprendizagem via Minimização Variacional Projetiva-Adaptativa não paramétrica (MVPAnP) que possui um caráter multifacetário, portanto, poderá ser aplicado em outros campo de estudo, que objetivam realizar a transferência de aprendizagem.

Finalmente, em vistas dos custos e da consequente inviabilidade orçamentária para aquisição da instrumentação da medição do STI, o modelo

proposto poderá ser utilizado como uma ferramenta de suporte para uma avaliação diagnóstica do STI em salas de aula, usando apenas a instrumentação que é utilizada para as medições.

5.2 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Ao tomar como base o desenvolvimento desse trabalho, sugere-se algumas 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 ruído de fundo de arquitetura estrutura, uma vez que nesse trabalho foram somente utilizados dados de medições de RIR das salas de aula da Universidade Federal do Paraná;
- Desenvolver o uso de algoritmo generativos para a criação e 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.

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.

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.

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.

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.

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, 2017. p. 5220-5224.

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**. 2018. p. 107-117.

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**. IEEE, 2007. p. 159-162.

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, 2017. p. 1257-1261.

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**. 2013. p. 2200-2207.

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

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, 2016. p. 015024.

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.

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.

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. **Sensor**s, 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**, 2014. p. 360-366.

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. Dec, 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, Ronald 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**. 2020. p. 1289-1299.

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, 2018. p. 591-595.

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

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.

STORN, R.; PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. **Journal of global optimization**, v. 11, n. 4, p. 341-359, 1997.

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.

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.; SUMMERS, J. E. Auralization: Fundamentals of acoustics, modelling, simulation, algorithms, and acoustic virtual reality. **The Journal of the Acoustical Society of America**, v. 123, n. 6, p. 4028, 2008.

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: ImprovingTransferability of Unsupervised DomainAdaptation by Gaussian-guided LatentAlignment. **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, 2019. p. 183-194.

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, 2014. p. 197-205.

ZHAI, S. et al. Doubly convolutional neural networks. **In: Advances in neural information processing systems**. 2016. p. 1082-1090.

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**, 2020.

ANEXO – CÓDIGO FONTE

SUMÁRIO DO CÓDIGO FONTE

Autor: *Eriberto Oliveira do Nascimento*
Data de criação: 17/02/2019
Última modificação: 07/05/2023
Descrição: Rede Neural Convolutional Variacional AutoCodificadora (CVAE) e Transferência de Aprendizagem Profunda com função de perda personalizada.
Descrição Longa: Esse é o código desenvolvimento para a tese de doutorado - DESENVOLVIMENTO DA TRANSFERÊNCIA DE APRENDIZAGEM VIA REDES NEURAIS ARTIFICIAIS PROFUNDAS NA MODELAGEM DO ÍNDICE DE TRANSMISSÃO DA FALA

Author: *Eriberto Oliveira do Nascimento*
Date created: 2019/17/02
Last modified: 2023/05/01
Description: Convolutional Variational AutoEncoder (CVAE) and Deep Transfer Learning with a custom loss function.
Long Description: This is the development code for the doctoral thesis - DEVELOPMENT OF DEEP TRANSFER LEARNING VIA ARTIFICIAL NEURAL NETWORKS IN MODELING THE INDEX OF SPEECH TRANSMISSION.

```
"""### 1 - Setup"""
#from __future__ import print_function
# -*- coding: utf-8 -*-

import sys
import os
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from scipy import stats

from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder

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

# prerequisites for the CVAE
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
from keras.datasets import mnist
from keras.layers import Input, Dense, Lambda
from keras.models import Model
from keras import backend as K
from keras import objectives
from scipy.stats import norm

import pandas as pd
!pip install acoustics
```

1 - Geração sintética da Resposta Impulsiva de Sala

1 - Synthetic room impulsive (RIR) response generation

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

"""
This example creates a room with reverberation time specified by inverting Sabine's formula.
This results in a reverberation time slightly longer than desired.
The simulation is pure image source method.
"""
import argparse

import matplotlib.pyplot as plt
import numpy as np
from scipy.io import wavfile
import librosa
# Uncoment this module to run it localy
# import pyroomacoustics as pra
import random
import time

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()

        # The desired reverberation time and dimensions of the room
        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)

        # import a mono wavfile as the source signal
        # the sampling frequency should match that of the room
        fs, audio = wavfile.read(signal_excitation)

        # We invert Sabine's formula to obtain the parameters for the ISM simulator
        e_absorption, max_order = pra.inverse_sabine(rt60_tgt, room_dim)

        # Create the room
        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()

# place the source in the room
source_pos = np.array(room_dim)/2;
source_pos[2] = 1.5
source_pos = source_pos.tolist()

# print(source_pos)

# source_pos = [2,2,1.5]
room.add_source(source_pos, signal=audio, delay=0.5)

# define the locations of the microphones

mic1 =   np.round( np.array(source_pos )+ np.array( [source_pos[0]*0.4, 0, 0]), 2)
# mic2 =   np.round(np.array(source_pos )+ np.array( [-source_pos[0]/2.5, 0, 0]), 2)
# mic3 =   np.round( np.array(source_pos )+ np.array( [0, source_pos[1]/2.5, 0]), 2)
mic4 =   np.round( np.array(source_pos )+ np.array( [0, -source_pos[1]/2.5, 0]), 2 )

mic1 = mic1.tolist()
# mic2 = mic2.tolist()
# mic3 = mic3.tolist()
# mic4 = mic4.tolist()

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


mic_locs = np.c_[
    mic1, mic4,
]

"""
 mic_locs = np.c_[
     [1,1,0.5], [1, 0.5, 2],  # mic 1  # mic 2
 ]
"""

# finally place the array in the room
room.add_microphone_array(mic_locs)

# Run the simulation (this will also build the RIR automatically)
room.simulate()

"""
```

```
        room.mic_array.to_wav(
            'Excitacao23.wav',
            norm=True,
           bitdepth=np.int16,
        )
        """
        # measure the reverberation time
        rt60 = room.measure_rt60()
        print("The desired RT60 was {}".format(rt60_tgt))
        print("The measured RT60 is {}".format(rt60[1, 0]))

        #Create a plot
        plt.figure()

        # plot one of the RIR. both can also be plotted using 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]")

        # plot signal at microphone 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()

        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 represents the total numbers of RIR generated
    while (i < 2):
        try:
            RIR_generator('Excitacao.wav')
        except Exception:
            # In any case of error during the function execution
            pass

        i = i +1
        print(i)

# Uncoment to run the generate_RIR() method that
# generate the RIR curves
# generate()
```

2 - Geração das amostras de ruído de fundo (BGN)

2 - Background noise (BGN) samples generation

```
# This is the link for all background noise sample applied on this thesis
```

3 - Cálculos de Densidade Espectral de Potência

3 - Calculations of Power Spectral Density (PSD)

```
# !pip install acoustics

from scipy.io import wavfile
import  scipy
import numpy as np
from matplotlib import pyplot as plt
from scipy import signal
import librosa
from scipy.fft import fft
from acoustics import Signal
import acoustics

"""
This example creates a room with reverberation time specified by inverting Sabine's formula.
This results in a reverberation time slightly longer than desired.
The simulation is pure image source method.
"""

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: power spectral density')
plt.xlabel('Frequency')
plt.ylabel('Power')
plt.tight_layout()
plt.show()

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()

# Acustica
# RIR
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()

# Nova tr
bands = acoustics.bands.octave(125, 2000)
b = acoustics.room.t60_impulse(file_name, bands, rt='t30')
print("Tempo de reverberacao")
print(b)
plt.show()

# Ruido
print('Adicao do ruido')

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

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

4 - Criação do Conjunto de Dados STI (Domínios de Origem e Alvo)

4 - Creation of the STI (Source-Target Domains) Dataset

```
import os
import glob
import acoustics
from acoustics import Signal
```

```
import numpy as np
import math
import librosa
import matplotlib.pyplot as plt
import random
import pandas as pd
import os
from scipy import signal
from numpy import savez_compressed

"""
- Simulated STI values calculated using Shoreder Equation
"""

def BGN_audios(folder):

    """
    Recursively converts WAV to spetogram arrays
    folder - folder to convert.
    """

    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):

    """
    Recursively converts WAV to spetogram arrays
    folder - folder to convert.
    """

    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 frequencias menores que 8000

    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 potencia
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]
Op_SL_octave = [61.4, 65.6, 62.3, 56.8, 51.3, 42.6, 33.6] # conforme a norma

# s =  os.path.basename(RIR_file)
# start = s.find("[") + len("["); end = s.find("]"); substring = s[start:end]
# chunks = substring.split(',')
# V = int(chunks[0]) * int(chunks[1]) * int(chunks[2])
# q = 2;

# Classroom noise-to-signal
# Classroom_SNR = 0.0032*V*q*(1/RT_octave)* np.exp(0.16/RT_octave) *
#  10**( 0.1*(Ln - Op_SL_octave) )

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])

# Inicialization

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))

# weighing
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)

            # mTF[m,octave] = (( 1 + (2*math.pi*modulation_freq[m]*RT_octave[octave]/ 13.82)**2 )**(-1/2)
)* (1 +  Classroom_SNR[octave])**(-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)

    #  return [id_number, np.round(STI,3), np.round(RT_octave,3), np.round( BGN_octave[1],3)]
    room_id = id_number  * np.ones(( len(xvals ), 1) )
    room_STI = np.round(STI,3) * np.ones(( len(xvals ), 1) )

    """
      df = pd.DataFrame({'Room_id': room_id,
                         'STI': room_STI,
                         'Frequ': np.transpose(xvals),
                         'PSD_RIR':  psd_RIR,
                         'PSD_BGN': psd_BGN
                         },
                        index = xvals
                        )
    """
    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"

# print("Number of simulated RIR .wav files")
# print(RIR_audios(folder_RIR))
# print("Number of meausured BGB .wav files")
# print(BGN_audios(folder_BGN))

BGN = BGN_audios(folder_BGN)
RIR = RIR_audios(folder_RIR)

# print( RIR_BGN_2_STI(RIR[10], BGN[6], 5) )
```

```
# Teste gerar STFT bacgroud
# audio2_STFT_spectogram(BGN[6],5)

# Teste CWT
# audio2_CWT_spectogram(RIR[20],5)

# 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)

    # print(A)
    # print(A.shape)

    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
            # 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 de Autoencoder Variacional (VAE) no Domínio de Origem.

5 - Variational Autoencoder (VAE) Training for the Source Domain.

```
# Load raw data
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 - Functions settings
"""

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()


def show_basic_dataframe_info(dataframe, preview_rows=20):

    """
    This function shows basic information for the given dataframe
    Args:
        dataframe: A Pandas DataFrame expected to contain data
        preview_rows: An integer value of how many rows to preview
    Returns:
        Nothing
    """

    # Shape and how many rows and columns
    print("Number of columns in the dataframe: %i" % (dataframe.shape[1]))
    print("Number of rows in the dataframe: %i\n" % (dataframe.shape[0]))
    print("First 20 rows of the dataframe:\n")
    # Show first 20 rows
    print(dataframe.head(preview_rows))
    print("\nDescription of dataframe:\n")
    # Describe dataset like mean, min, max, etc.
    # print(dataframe.describe())


def read_data(file_path):

    """
    This function is the ETL process form the STI values
    Args:
        file_path: URL pointing to the pickle file
    Returns:
        A pandas dataframe
    """

    df.rename(columns = {'c1': 'Room-id',
                         'c2': 'STI',
                         'c3': 'Freq - [Hz]',
                         'c4': 'PSD(RIR)',
```

```
                    'c5': 'PSD(BGN)' },
                    inplace=True)

    # This is very important otherwise the model will not fit and loss
    # will show up as NAN
    label_encoder = LabelEncoder()
    n_bins = 10 # number of classes

    # number of classes
    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()


def create_segments_and_labels(df, time_steps, step, label_name):
```

```
    """
    This function receives a dataframe and returns the reshaped segments
    of BGN, RIR data as well as the corresponding STI labels
    Args:
        df: Dataframe in the expected format
        time_steps: Integer value of the length of a segment that is created
    Returns:
        reshaped_segments
        labels:
    """

    N_FEATURES = 2
    # Number of steps to advance in each iteration (it should always
    # be equal to the time_steps in order to have no overlap between segments)
    # step = time_steps

    segments = []
    labels = []

    for i in range(0, len(df) - time_steps, step):
        xs = df['PSD(RIR)'].values[i: i + time_steps]
        ys = df['PSD(BGN)'].values[i: i + time_steps]
        xs = xs[1:]
        ys = ys[1:]
        # Retrieve the most often used label in this segment
        label = stats.mode(df[label_name][i: i + time_steps])[0][0]
        segments.append([xs, ys])
        labels.append(label)

    # Bring the segments into a better shape
    reshaped_segments = np.asarray(segments,
                                   dtype= np.float32).reshape(-1,
                                                              time_steps-1,
                                                              N_FEATURES)
    labels = np.asarray(labels)

    return reshaped_segments, labels

"""

## Create training database
"""

# Set some standard parameters upfront

pd.options.display.float_format = '{:.1f}'.format

# sns.set() # Default seaborn look and feel
# plt.style.use('ggplot')

print('keras version ', keras.__version__)

LABELS = ["1","2","3","4","5","6","7","8","9","10"]

# The number of steps within one frequenct range segment
TIME_PERIODS = len(np.arange(0,8001,10))
STEP_DISTANCE = TIME_PERIODS
```

```
print("\n--- Load, inspect and transform data ---\n")

print("Load data set from the npz")
read_data(df)

# Describe the data
show_basic_dataframe_info(df, 20)

(df['STI']).value_counts().plot(kind='bar',
                                title='Training Examples by STI classes')

plt.savefig('STI_distribution_source_domain.pdf')
# plt.show()

print("\n--- Reshape the data into segments ---\n")

# Differentiate between test set and training set
df_train = df[df['Room-id'] <= 25000]
df_test = df[df['Room-id'] > 25000]

# Normalize features for training data set

from sklearn.preprocessing import MinMaxScaler

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

# Now for the test
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

# Round in order to comply to NSNumber from RIR curvers
df_train = df_train.round({'PSD(RIR)': 6, 'PSD(BGN)': 6})
df_test =  df_test.round({'PSD(RIR)': 6, 'PSD(BGN)': 6})

# Reshape the training data into segments, so that
# they can be processed by the network
# Define column name of the label vector

LABEL = "STI"

x_train, y_train = create_segments_and_labels(df_train,
                                              TIME_PERIODS,
                                              STEP_DISTANCE,
                                              LABEL)
```

```
x_test, y_test = create_segments_and_labels(df_test,
                                            TIME_PERIODS,
                                            STEP_DISTANCE,
                                            LABEL)

print("\n--- Reshape data to be accepted by Keras ---\n")

# Inspect x data
print('x_train shape: ', x_train.shape)
print(x_train.shape[0], 'training samples')

# Inspect y data
print('y_train shape: ', y_train.shape)

print('y_test shape: ', y_test.shape)

# Set input_shape / reshape for Keras

print(" Verifying the training dataset dimensions")

print("Dimensoes antes")
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

# Input image dimensions
img_rows = 1600

# Channels go last for TensorFlow backend
print("Dimensão após restruturacao - treino")
x_train_reshaped = x_train.reshape(x_train.shape[0], img_rows)
print(x_train_reshaped.shape)

# Now verifies the test
print("Dimensão após restruturacao - teste")
x_test_reshaped = x_test.reshape(x_test.shape[0], img_rows)
print(x_test_reshaped.shape)

x_tr = x_train_reshaped
x_te = x_test_reshaped
```

6 5.1 - Estatísticas de treinamento para o domínio de origem

7 5.1 - Training Statistics for the Source Domain

```
### Fitting
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats

# Extract the column from the dataframe
# and convert it to a numpy array
data = (df['STI']/10).values

# Fit a Gaussian distribution to the data
mu, std = stats.norm.fit(data)

# Plot the histogram
```

```
plt.hist(data, bins=20, density=True, alpha=0.6, color='b')

# Add a title and labels
plt.xlabel('STI')
plt.ylabel('Função densidade de probabilidade')

# Show the plot
print(mu, std)

# Plot the histogram
plt.hist(data, bins=20, density=True, alpha=0.6, color='b')

# Plot the Gaussian fit
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, mu, std)

#plt.plot(x, c, 'r', linewidth=2)
# Add a title and labels
plt.xlabel('STI')
plt.ylabel('Função densidade de probabilidade')

ax2 = plt.twinx()

# plot the data for the second y-axis
ax2.plot(x, p, 'r--')
ax2.set_ylabel('Função densidade acumulada', color='r')

# show the plot
plt.show()


# Show the plot
print(mu, std)

# Plot the histogram
plt.hist(data, bins=20, density=True, alpha=0.6, color='b')

# Plot the Gaussian fit
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
c = stats.norm.cdf(x, mu, std)

#plt.plot(x, c, 'r', linewidth=2)
# Add a title and labels
plt.xlabel('STI')
plt.ylabel('Função densidade de probabilidade')

ax2 = plt.twinx()

# plot the data for the second y-axis
ax2.plot(x, c, 'r--')
ax2.set_ylabel('Função densidade acumuldada', color='r')

# show the plot
plt.show()
```

```
# Show the plot
print(mu, std)
```

8 5.2 - Treinamento da rede autoencoder variacional

9 5.2 - VAE training

```
"""## 3 - Reescaling the data for the VAE traning"""

print(" Build the VAE models and set is parameters ")
batch_size, n_epoch = 1, 5
n_hidden, z_dim = 64, 2 # Here changes the z-dimension (latent space)

# 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)

# sampling function
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

# build model
vae = Model(x, y)
vae.add_loss(vae_loss)

# Add metrics
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()

# train
vae.fit(x_tr,
        shuffle=True,
        epochs=n_epoch,
```

```
        batch_size=batch_size,
        validation_data=(x_te, None), verbose=1)

# build encoder
encoder = Model(x, mu)
encoder.summary()
```

5.3 - Valores da função perda da rede VAE

5.3 - VAE losses values

```
vae.history

# list all data in history
print(vae.history.history.keys())

print(vae.history.history['loss'])

# Extract the loss values from the history and plot it
loss_values = vae.history.history['loss']

# Plot the loss values over time
plt.plot(loss_values)
plt.xlabel('Época')
plt.ylabel('Função de perda')
plt.show()

#List of all possible metrics to plot:

'''
dict_keys(['loss', 'reconstruction_loss',
           'kl_loss', 'vae_loss',
           'val_loss', 'val_reconstruction_loss',
           'val_kl_loss', 'val_vae_loss'])
'''

# Extract the loss values from the history
component1_values = vae.history.history['reconstruction_loss']
component2_values = vae.history.history['val_reconstruction_loss']

# Plot the loss values over time
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()

# Extract the KL loss values from the history and plot it
component1_values = vae.history.history['kl_loss']
component2_values = vae.history.history['val_kl_loss']

# Plot the loss values over time
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()
```

```
# Extract the loss values from the history
component1_values = vae.history.history['vae_loss']
component2_values = vae.history.history['val_vae_loss']

# Plot the loss values over time
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()
```

6 - Otimização de hiperparâmetros (HPO) do modelo VAE

6 - Hyperparameter Optimization (HPO) of the VAE model

```
# You can use Matplotlib instead of Plotly for visualization by
#  simply replacing `optuna.visualization` with
# `optuna.visualization.matplotlib` in the following examples.

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

SEED = 42

np.random.seed(SEED)

from keras.backend import clear_session
from keras.utils.vis_utils import plot_model
from sklearn.metrics import mean_squared_error

def start_Autoencoder(features,trials,  plot_graph = False):

  # Variational Autoencoder structure
  def create_model(activation, layers_num, zdim_num, kernel_initializer):
    clear_session()

    batch_size, n_epoch = 1, 1
    n_hidden, z_dim = layers_num, zdim_num
    # Here change the dimension (Model hyperparameter)

    # encoder
    x = Input(shape=(x_tr.shape[1:]))
    x_encoded = Dense(n_hidden,
                      activation=activation,
                      kernel_initializer=kernel_initializer)(x)
    x_encoded = Dense(n_hidden//2, activation=activation)(x_encoded)

    mu = Dense(z_dim)(x_encoded)
    log_var = Dense(z_dim)(x_encoded)
```

```
    # sampling function
    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

    # build model
    vae = Model(x, y)

    vae.add_loss(vae_loss)
    # Add metrics
    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

# Objective function to optimize by OPTUNA module

def objective(trial):
    activation = trial.suggest_categorical("activation",
                                    ["relu", "sigmoid", "swish"])
    layers_num = trial.suggest_int("layers_num", 2,64)
    zdim_num = trial.suggest_int("dim_latent_z", 2,128)
    if (activation == "relu"):
      model = create_model(activation,
                        layers_num,
                        zdim_num,
                        kernel_initializer="HeUniform")
    else:
      model = create_model(activation,
                        layers_num,
                        zdim_num,
                        kernel_initializer="GlorotUniform")

    model.compile(optimizer='rmsprop')

    # Implement early stopping criterion.
    # Training process stops when there is no improvement during 50 iterations

    callback = keras.callbacks.EarlyStopping(monitor='loss', patience=50)
    history = model.fit(features,
```

```
                    features,
                    batch_size = 1,
                    epochs=1,
                    callbacks = [callback],
                    verbose = 0)

    return history.history["loss"][-1]

  study = optuna.create_study(direction='minimize')
  study.optimize(objective, n_trials=trials)

  # Create final model with the best hyperparams
  print('Best hyperparams found by Optuna: \n', study.best_params)
  if (study.best_params['activation'] == "relu"):
    model = create_model(study.best_params['activation'],
                     int(study.best_params['layers_num']),
                     int(study.best_params['dim_latent_z']),
                     kernel_initializer="HeUniform")
  else:
    model = create_model(study.best_params['activation'],
                    int(study.best_params['layers_num']),
                    int(study.best_params['dim_latent_z']),
                    kernel_initializer="GlorotUniform")

  model.compile(optimizer='rmsprop')
  model.summary()

  # Implement early stopping criterion.
  # Training process stops when there is no improvement during a certatin number
  # of iterations

  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)

  # Result evaluation
  print(f'RMSE Autoencoder: {np.sqrt(mean_squared_error(features, result))}')
  print('')

  # Following values are returned: extracted_f || MSE || OPTUNA best hyperparams

  return mean_squared_error(features, result), study.best_params, study


# Run the HPO on the VAE

Acoder_MSE, Acoder_hyperparams, Study = start_Autoencoder(features = x_tr,
                              trials = 5,
                              plot_graph=True)

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)
```

7 - Visualização dos Embeddings VAE

7 - Visualization of the VAE Embeddings

```
import matplotlib

cmap = matplotlib.cm.get_cmap('Spectral')

cmap

def plot_representation(features, labels, rep_type):
    fig, axes = plt.subplots(2, 3, figsize=(12,6))
    fig.suptitle('Scatter plot: ' + rep_type + ' 4 features representation', fontsize=16)

    for i, pair in enumerate(combinations([0,1,2,3], 2)):
        for label, color in zip([1,2,3,4,5,6,7,8,9], ['blue', 'green',
                                                      'black', 'yellow',
                                                      'brown', 'orange',
                                                      'red', 'gray',
                                                      'olive','gold']):

            axes[int(i / 3), int(i % 3)].scatter(features[labels == label, pair[0]],
                                                 features[labels == label, pair[1]],
                                                 c=color,
                                                 alpha=0.5,
                                                 label = label)

            axes[int(i / 3), int(i % 3)].set_xlabel(f'Feature {pair[0]}')
            axes[int(i / 3), int(i % 3)].set_ylabel(f'Feature {pair[1]}')
            axes[int(i / 3), int(i % 3)].legend()

    plt.tight_layout()

from sklearn.manifold import TSNE, Isomap
from sklearn.decomposition import PCA
from itertools import combinations

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)

# Visualize the results of the dimensionaly reduction
plot_representation(PCA_representation, y_train, 'PCA')
```

8 - Salvando o modelo do codificador em uma extensão de arquivo .h5

8 - Saving the Encoder Model to a .h5 File Extension

```
# Save weights approaches
```

```
#vae.save_weights('my_vae_weights.h5')
#encoder.save("my_h5_model.h5")

# Load weights
# Calling `save('my_model.h5')` creates a h5 file `my_model.h5`.
#encoder.save("my_h5_model.h5")

# It can be used to reconstruct the model identically.
#reconstructed_model = keras.models.load_model("my_h5_model.h5")

# Let's check: (Test to verify if the model was reload properly)
#np.testing.assert_allclose(
#    encoder.predict(test_input), reconstructed_model.predict(test_input)
#)

# The reconstructed model is already compiled and has retained the optimizer
# state, so training can resume:
#reconstructed_model.fit(test_input, test_target)

# Calling `save('my_model.h5')` creates a h5 file `my_model.h5`.
encoder.save("vae_encoder.h5")

# It can be used to reconstruct the model identically.

# Load the saved model and make predicitions
reconstructed_model = keras.models.load_model("vae_encoder.h5")
x_te_latent_recons = reconstructed_model.predict(x_te, batch_size=1)

# Compare with the orginal encoder model
x_te_latent_origin = encoder.predict(x_te, batch_size=batch_size)

# Compare the results
x_te_latent_origin == x_te_latent_recons
```

8.2 - Salvando o modelo do codificador como um objeto gráfico do TensorFlow

8.2 Saving the Encoder Model as a TensorFlow Graph Object

```
import tensorflow as tf

# Save the model
# encoder.save("my_model.model")
encoder.save("my_model.model")

# Load the saved model
tensorflow_graph = tf.saved_model.load("./my_model.model")

# Make predictions
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()

# predicted
# Compare the results of the saved model and the original model in memory
np.round(z_latent,4) == np.round(x_te_latent_origin,4)
```

8.3 - Salvando as previsões do conjunto de dados de treinamento (x_tr_latent) no espaço codificado para modelagem KCPR

8.3 - Saving the Training Dataset Predictions (x_tr_latent) to the Encoded Space for KCPR Modeling

```
# Important: Generate the z latent encoding  with 128 dimensions, then z = 12

# x_tr
x_tr_latent = encoder.predict(x_tr, batch_size=batch_size)

# Make a pandas dataframe of the latent space and export the data
x_tr_latent.shape

# Create the padnas dataframe
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)

# print(classes)
dataset_high_z_dim.to_csv('dataset_high_z_dim.csv')
```

9 - Plotando a visualização do espaço do kernel 2D

9 - Plotting 2D kernel space visualization

```
#Inspecting the latent space varialbes (z1 and z2)
x_tr_latent
```

9.1 - Plotando a representação do espaço latente (z = 2) do treinamento (conjunto de dados de origem)

9.1 Plotting the Latent Space Representation (z = 2) of the Training (Source Dataset)

```
# Plot
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 (domínio) ")
plt.xlabel('Dimensão - z1')
plt.ylabel('Dimensão - z2')
#ax.add_artist(legend1)

plt.show()
```

9.2 Plotando a representação do espaço latente (z = 2) do teste (conjunto de dados de origem)

9.2 Plotting the Latent Space Representation (z = 2) of the Testing (Source Dataset)

```
# Make the prediction of the latent space
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()

# produce a legend with the unique colors from the scatter
plt.show()
```

9.3 - Reconstruindo o espaço latente VAE via codificador de mapas embuídos (Domínio de origen)

9.3 - Reconstructing VAE Latent Space via Encoder Embedding (Source Domain)

```
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

9.3.1 - Visualization of Latent Space Histogram Inference

```
plt.figure(figsize=(6, 6))
dataset['z1'].plot(kind="hist", bins=20, density=True, alpha=0.6 )
dataset['STI'].plot(kind="hist", bins=20, density=True, alpha=0.6)
dataset['z2'].plot(kind="hist", bins=20, density=True, alpha=0.6)

plt.legend(title="STI - Dados de validação (domínio)")
plt.xlabel('Dimensão latente')
plt.ylabel('Função densidade de probabilidade')
plt.show()
```

9.3.2 - IA Explicativa (EAX) - Plotando a reconstrução do modelo codificado em 2 dimensões

9.3.2 - Explainable AI (EAX) - Plotting the Encoded Model Reconstruction in 2-Dimensions

```
# build decoder
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)

# generator.summary()
# display a 2D manifold of the digits

n = 15 # figure with 15x15 dimensional size
digit_size1 = 40
digit_size2 = 40
figure = np.zeros((digit_size1 * n, digit_size2 * n))

grid_x = norm.ppf(np.linspace(0.05, 0.95, n))
grid_y = norm.ppf(np.linspace(0.05, 0.95, n))

for i, yi in enumerate(grid_x):
    for j, xi in enumerate(grid_y):
        z_sample = np.array([[xi, yi]])
        x_decoded = generator.predict(z_sample)
        digit = x_decoded[0].reshape(digit_size1, digit_size2)
        figure[i * digit_size1: (i + 1) * digit_size1,
               j * digit_size2: (j + 1) * digit_size2] = digit

plt.figure(figsize=(10, 10))
```

```
plt.xlabel('Dimensão - z1')
plt.ylabel('Dimensão - z2')
plt.imshow(figure, cmap='jet')
plt.show()
```

9.3.3 PCA aplicado ao espaço latente durante o treinamento de VAE: z Dim = 2

9.3.3 - PCA Applied to the Latent Space During VAE Training: z Dimension = 2

```
x_te_latent_recons.shape

y_te.min()

# Reload data
# PCA_transformer = PCA(n_components=4)
# x_train_reshaped_TSE = np.asarray(x_te_latent_recons, dtype='float64')
# PCA_representation = PCA_transformer.fit_transform(x_train_reshaped_TSE)

# Visualize results
# plot_representation(PCA_representation, y_te, 'PCA')
```

9.3.4 - Visualização da estimativa de densidade do kernel do domínio de origem

9.3.4 - Visualization of the Kernel Density Estimation of the source domain

```
###
is_STI_1 =  dataset['STI'] == 1
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.1')
plt.show()
###
###
is_STI_1 =  dataset['STI'] == 2
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.2')
plt.show()
###
is_STI_1 =  dataset['STI'] == 3
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.3')
plt.show()
###
is_STI_1 =  dataset['STI'] == 4
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
```

```
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.4')
plt.show()
####
is_STI_1 =  dataset['STI'] == 5
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.5')
plt.show()
####
is_STI_1 =  dataset['STI'] == 6
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.6')
plt.show()
####
is_STI_1 =  dataset['STI'] == 7
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.7')
plt.show()
####
is_STI_1 =  dataset['STI'] == 8
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.8')
plt.show()
####
is_STI_1 =  dataset['STI'] == 9
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.9')
plt.show()
```

9.3.5 - Mapeamento do conjunto de origem

9.3.5 - - Source Domain Mapping

```
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

9.3.6 - 2D KDE Visualization of the Bidimensional Encoded Space and STI Values for the Source-Target Domains

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import kde

######## create data
plt.figure(figsize=(6, 6))
is_STI_1 =  dataset['STI'] == 1
is_STI_1_data = dataset[is_STI_1]

x = is_STI_1_data['z1']
y = is_STI_1_data['z2']

# Evaluate a gaussian kde on a regular grid of nbins x nbins over data extents
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()]))

# Make the plot
plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='auto')

plt.xlabel('Dimensão latente - z1')
plt.ylabel('Dimensão latente - z1')
plt.title('STI = 0.1')
plt.show()

######## create data
plt.figure(figsize=(6, 6))
is_STI_1 =  dataset['STI'] == 3
is_STI_1_data = dataset[is_STI_1]

x = is_STI_1_data['z1']
y = is_STI_1_data['z2']

# Evaluate a gaussian kde on a regular grid of nbins x nbins over data extents
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()]))

# Make the plot
plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='auto')

plt.xlabel('Dimensão latente - z1')
plt.ylabel('Dimensão latente - z1')
plt.title('STI = 0.3')
plt.show()
```

```
######## create data
plt.figure(figsize=(6, 6))
is_STI_1 =  dataset['STI'] == 6
is_STI_1_data = dataset[is_STI_1]

x = is_STI_1_data['z1']
y = is_STI_1_data['z2']

# Evaluate a gaussian kde on a regular grid of nbins x nbins over data extents
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()]))

# Make the plot
plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='auto')

plt.xlabel('Dimensão latente - z1')
plt.ylabel('Dimensão latente - z1')
plt.title('STI = 0.6')
plt.show()
#
######## create data
plt.figure(figsize=(6, 6))
is_STI_1 =  dataset['STI'] == 9
is_STI_1_data = dataset[is_STI_1]

x = is_STI_1_data['z1']
y = is_STI_1_data['z2']

# Evaluate a gaussian kde on a regular grid of nbins x nbins over data extents
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()]))

# Make the plot
plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='auto')

plt.xlabel('Dimensão latente - z1')
plt.ylabel('Dimensão latente - z1')
plt.title('STI = 0.9')
plt.show()

# Change color palette
#plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='auto', cmap=plt.cm.Greens_r)
#plt.show()
```

10 - KCPR e outros métodos de regressão multivariada

10 - KCPR and Other Multivariate Regression Methods

Ridge Regression, PCR, KCPR, and PLSR: Multivariate Regression Methods

10.1 - Rigde Regression

```
from numpy import arange
from pandas import read_csv
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedKFold
from sklearn.linear_model import Ridge

# load the dataset
X, y = dataset[['z1','z2']].values, dataset['STI'].values

# define model
model = Ridge()
# define model evaluation method
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)

# define grid
grid = dict()
grid['alpha'] = arange(0, 1, 0.01)

# define search
search = GridSearchCV(model, grid,
                      scoring='neg_mean_absolute_error',
                      cv=cv,
                      n_jobs=-1)

# perform the search
results = search.fit(X, y)
# summarize
print('MAE: %.3f' % results.best_score_)
print('Config: %s' % results.best_params_)
```

10.2 - PCR - KCPR - PLSR

```
# Since we are using unbalaced data (STI values), we must first balance out the data
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
over_sampler = RandomOverSampler(random_state=42)
X_res, y_res = over_sampler.fit_resample(X, y)

print(f"Training target statistics: {Counter(y_res)}")
print(f"Testing target statistics: {Counter(y_test)}")
```

10.3 - Comparação e Benchmarking de Modelos de Regressão Multivariada

10.3 - Comparison and Benchmarking of Multivariate Regression Models

```
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVR
from sklearn.cross_decomposition import PLSRegression
from sklearn.decomposition import KernelPCA

from imblearn.over_sampling import RandomOverSampler
from collections import Counter
over_sampler = RandomOverSampler(random_state=42)

X, y = dataset[['z1','z2']].values, dataset['STI'].values
X_res, y_res = over_sampler.fit_resample(X, y)

print(f"Training target statistics: {Counter(y_res)}")
```

```
print(f"Testing target statistics: {Counter(y_test)}")

# X, y = dataset[['z1','z2']].values, dataset['STI'].values

X_train, X_test, y_train, y_test = train_test_split(X, y)


pcr = make_pipeline(StandardScaler(),
                    PCA(n_components=2),
                    LinearRegression())

pcr.fit(X_train, y_train)

pca = pcr.named_steps["pca"]  # retrieve the PCA step of the pipeline

kpcr = make_pipeline(StandardScaler(),
                     KernelPCA(kernel="rbf", n_components=2),
                     LinearRegression()
                    )

kpcr.fit(X_train, y_train)
# kpca = kpcr.named_steps["KernelPCA"]  # retrieve the PCA step of the pipeline

pls = PLSRegression(n_components=1)
pls.fit(X_train, y_train)

svr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2))
svr.fit(X_train, y_train)

fig, axes = plt.subplots(1, 4, figsize=(10, 3))

axes[0].scatter(
    y_test, pcr.predict(X_test), alpha=0.3, label="predictions"
)
axes[0].set(xlabel="STI augmented", ylabel="y", title="PCR")

axes[1].scatter(
    y_test, pls.predict(X_test), alpha=0.3, label="predictions"
)

axes[1].set(xlabel="STI augmented", ylabel="y", title="PLSR")

axes[2].scatter(
    y_test, kpcr.predict(X_test), alpha=0.3, label="predictions"
)
axes[2].set(xlabel="STI augmented", ylabel="y", title="KPCR")

axes[3].scatter(
    y_test, svr.predict(X_test), alpha=0.3, label="predictions"
)
axes[3].set(xlabel="STI augmented", ylabel="y", title="SVR")

for ax in axes.flat: #this will iterate over all 6 axes
    ax.set_xlim(-1, 10)
    ax.set_ylim(-1, 10)

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

10.4 - Modelagem da Regressão de Kernel (KCPR)

10.4 - KCPR Modeling

```
from sklearn.gaussian_process.kernels import ConstantKernel, RBF, DotProduct
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from numpy import linalg as LA

# https://web.mit.edu/modernml/course/lectures/MLClassLecture3.pdf (Pag. 5)
# kernel = RBF()
# RBF_KERNEL = kernel.__call__(X)
# kernel = ConstantKernel()
# kernel.__call__(X)
# kernel = DotProduct()
# kernel.__call__(X)

# Load the dataset
df_KPCR  = pd.read_csv('/kaggle/input/sti-prediction/dataset_high_z_dim_to_KCPR.csv')
df_KPCR.dropna(inplace=True)

# center the values in each column of the DataFrame
# X = df[['z1','z2']].values
# y = df['STI'].values

df_KPCR = df_KPCR.groupby(by=["classes"], dropna=True).mean()
df_KPCR = df_KPCR.drop(columns=["Unnamed: 0"])

# Mean centraing
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()

y_KPCR
X_KPCR.shape
X_KPCR

# Get the kernel and calculte the respective transformation
kernel = RBF(0.2)
K = kernel.__call__(X_KPCR)

# centralize the kenerl
'''
N = K.shape[0]
one_n = np.ones((N,N)) / N
K = K - one_n.dot(K) - K.dot(one_n) + one_n.dot(K).dot(one_n)
'''

# Verify if the kernel is centralized by the mean
# print(K.mean(axis=1))

# Calcuting the eigedecompostion
# eigenValues, eigenVectors = LA.eig(np.transpose(K) * K)
```

```
eigenValues, eigenVectors = LA.eig(K)

# Verirify if the kernel othornomal
# abs(eigenVectors)

Z = K  * eigenVectors
# Z = K  * abs(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.show()

#M aking the model to predict function
# https://web.mit.edu/modernml/course/lectures/MLClassLecture3.pdf

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()
y_hat
```

10.5 - Persistência do modelo KCPR e SVR para ser usada como uma função de perda personalizada no modelo MVPAnP

10.5 - KCPR and SVR Model persistence to be used as a custom loss function in the model MVPAnP Model

```
from joblib import dump, load
dump(svr, 'srv_reg.joblib')

svr_persited  = load('srv_reg.joblib')

svr_persited.predict(X_res[0:1])
```

11 - Metodologia de Transferência de Aprendizagem

11 - Transfer Learning Methodology

11.1 - Carregar os dados do domínio de destino

11.1 - Load the Target Domain Data

```
## Loader function of test/validation dataset for transfer learning

# Load data
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 - Functions settings
"""
```

```
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()


def show_basic_dataframe_info(dataframe,
                              preview_rows=20):

    """
    This function shows basic information for the given dataframe
    Args:
        dataframe: A Pandas DataFrame expected to contain data
        preview_rows: An integer value of how many rows to preview
    Returns:
        Nothing
    """

    # Shape and how many rows and columns
    print("Number of columns in the dataframe: %i" % (dataframe.shape[1]))
    print("Number of rows in the dataframe: %i\n" % (dataframe.shape[0]))
    print("First 20 rows of the dataframe:\n")
    # Show first 20 rows
    print(dataframe.head(preview_rows))
    print("\nDescription of dataframe:\n")
    # Describe dataset like mean, min, max, etc.
    # print(dataframe.describe())


def read_data(file_path):

    """
    This function reads the data from a file
    Args:
        file_path: URL pointing to the pickle file
    Returns:
        A pandas dataframe
    """
```

```
    df.rename(columns = {'c1': 'Room-id', 'c2': 'STI',
                         'c3': 'Freq - [Hz]', 'c4': 'PSD(RIR)',
                         'c5': 'PSD(BGN)' },  inplace = True)

    # This is very important otherwise the model will not fit and loss
    # will show up as NAN

    label_encoder = LabelEncoder()
    n_bins = 10 # number of classes

   # 10 number of classes accoring to the STI range (0,1)
    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


# Not used right now
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()


def create_segments_and_labels(df, time_steps, step, label_name):
```

```
    """
    This function receives a dataframe and returns the reshaped segments
    of BGN, RIR data as well as the corresponding STI labels
    Args:
        df: Dataframe in the expected format
        time_steps: Integer value of the length of a segment that is created
    Returns:
        reshaped_segments
        labels:
    """

    N_FEATURES = 2

    # Number of steps to advance in each iteration (for me, it should always
    # be equal to the time_steps in order to have no overlap between segments)
    # step = time_steps

    segments = []
    labels = []

    for i in range(0, len(df) - time_steps, step):
        xs = df['PSD(RIR)'].values[i: i + time_steps]
        ys = df['PSD(BGN)'].values[i: i + time_steps]
        xs = xs[1:]
        ys = ys[1:]
        # Retrieve the most often used label in this segment
        label = stats.mode(df[label_name][i: i + time_steps])[0][0]
        segments.append([xs, ys])
        labels.append(label)

    # Bring the segments into a better shape
    reshaped_segments = np.asarray(segments, dtype= np.float32) \
                        .reshape(-1, time_steps-1, N_FEATURES)
    labels = np.asarray(labels)

    return reshaped_segments, labels

"""

## Create training database
"""

# Set some standard parameters upfront
pd.options.display.float_format = '{:.1f}'.format
# sns.set() # Default seaborn look and feel
# plt.style.use('ggplot')
print('keras version ', keras.__version__)

LABELS = ["1","2","3","4","5","6 ","7","8","9","10"]

# The number of steps within one time segment
TIME_PERIODS = len(np.arange(0,8001,10))
STEP_DISTANCE = TIME_PERIODS

print("\n--- Load, inspect and transform data ---\n")

print("Load data set from the npz")
read_data(df)
```

```
# Describe the data
show_basic_dataframe_info(df, 20)

# Total number of rooms
print("\n--- Calculating total number of classrooms ---\n")
num_rooms = len(df) / TIME_PERIODS

df['STI'].value_counts().plot(kind='bar', title='Training Examples by STI classes')
plt.savefig('STI_distribution_target_domain.pdf')
#plt.show()

print("\n--- Reshape the data into segments ---\n")

# Differentiate between test set and training set
df_train = df[df['Room-id'] <= 0.7* num_rooms]
df_test = df[df['Room-id'] > 0.7* num_rooms]

# Normalize features for training data set
from sklearn.preprocessing import MinMaxScaler

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

# Test dataset
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

# Round in order to comply to NSNumber
df_train = df_train.round({'PSD(RIR)': 6, 'PSD(BGN)': 6})
df_test =  df_test.round({'PSD(RIR)': 6, 'PSD(BGN)': 6})

# Reshape the training data into segments, so that they can be processed by
# the network
# Define column name of the label vector

LABEL = "STI"
x_train, y_train = create_segments_and_labels(df_train,
                                              TIME_PERIODS,
                                              STEP_DISTANCE,
                                              LABEL)

x_test, y_test = create_segments_and_labels(df_test,
                                            TIME_PERIODS,
                                            STEP_DISTANCE,
                                            LABEL)
```

```
print("\n--- Reshape data to be accepted by Keras ---\n")

# Inspect x data
print('x_train shape: ', x_train.shape)
print(x_train.shape[0], 'training samples')

# Inspect y data
print('y_train shape: ', y_train.shape)

print('y_test shape: ', y_test.shape)

# Set input_shape / reshape for Keras

print(" Verifying the trainign dataset dimensions")

print("Dimensoes antes")
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

# Input image dimensions
img_rows = 1600
# len(np.arange(0,8001,10)) aproximação 800 TR and 800 TO BGN

# Channels go last for TensorFlow backend
print("Dimensão após restruturacao - treino")
x_train_reshaped = x_train.reshape(x_train.shape[0], img_rows)
print(x_train_reshaped.shape)

# Agora fazer o teste
print("Dimensão após restruturacao - teste")
x_test_reshaped = x_test.reshape(x_test.shape[0], img_rows)
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

11.2 - Target Domain Data Distribution Statistics

```
### Fitting
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats

# Extract the column from the dataframe and convert it to a numpy array
data = (df['STI']/10).values

# Fit a Gaussian distribution to the data
mu, std = stats.norm.fit(data)

# Plot the histogram
plt.hist(data, bins=20, density=True, alpha=0.6, color='r')

# Add a title and labels
plt.xlabel('STI')
plt.ylabel('Função densidade de probabilidade')
```

```
# Show the plot
print(mu, std)

# Plot the histogram
plt.hist(data, bins=20, density=True, alpha=0.6, color='r')

# Plot the Gaussian fit
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, mu, std)

# plt.plot(x, c, 'r', linewidth=2)
# Add a title and labels
plt.xlabel('STI')
plt.ylabel('Função densidade de probabilidade')

ax2 = plt.twinx()

# plot the data for the second y-axis
ax2.plot(x, p, 'r--')
ax2.set_ylabel('Função densidade acumuldada', color='b')

# show the plot
plt.show()


# Show the plot
print(mu, std)

# Plot the histogram
plt.hist(data, bins=20, density=True, alpha=0.6, color='r')

# Plot the Gaussian fit
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
c = stats.norm.cdf(x, mu, std)

# plt.plot(x, c, 'r', linewidth=2)
# Add a title and labels
plt.xlabel('STI')
plt.ylabel('Função densidade de probabilidade')

ax2 = plt.twinx()

# plot the data for the second y-axis
ax2.plot(x, c, 'r--')
ax2.set_ylabel('Função densidade acumulada', color='r')

# show the plot
plt.show()


# Show the plot
print(mu, std)

### Fitting
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
```

```
# Extract the column from the dataframe and convert it to a numpy array
data = df['STI'].values

# Fit a Gaussian distribution to the data
mu, std = stats.norm.fit(data)

# Plot the histogram
plt.hist(data, bins=25, density=True, alpha=0.6, color='b')

# Plot the Gaussian fit
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)

# Add a title and labels
plt.xlabel('STI')
plt.ylabel('Probability Density')

# Show the plot
print(mu, std)

x_tr.shape

y_train.shape
```

12 - Reconstruindo o espaço latente VAE para o domínio de destino por meio do modelo de codificador treinado no domínio de origem

12 - Reconstructing the VAE Latent Space for the Target Domain via the Encoder Model Trained on the Source Domain

```
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

12.1 - Plotting the Latent Space Representation (z = 2) of the Testing Data in the Target Dataset

```
# Latent Space obtined from the target domain
# Plot the target domain (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')
#ax.add_artist(legend1)
```

```
plt.show()
```

12.1.1 - Visualização da Estimativa de Densidade do Kernel (KDE) para o Domínio de Destino

12.1.1 - Visualization of Kernel Density Estimation (KDE) for the Target Domain

```
###
is_STI_1 =  dataset['STI'] == 1
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.1')
plt.show()
###
###
is_STI_1 =  dataset['STI'] == 2
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.2')
plt.show()
###
is_STI_1 =  dataset['STI'] == 3
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.3')
plt.show()
###
is_STI_1 =  dataset['STI'] == 4
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.4')
plt.show()
####
is_STI_1 =  dataset['STI'] == 5
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.5')
plt.show()
####
is_STI_1 =  dataset['STI'] == 6
```

```
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.6')
plt.show()
####
is_STI_1 =  dataset['STI'] == 7
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.7')
plt.show()
####
is_STI_1 =  dataset['STI'] == 8
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.8')
plt.show()
####
is_STI_1 =  dataset['STI'] == 9
is_STI_1_data = dataset[is_STI_1]
#print(is_STI_1_data)
is_STI_1_data = is_STI_1_data[['z1','z2']]
ax = is_STI_1_data.plot.kde()
plt.xlabel('Dimensão latente - z')
plt.ylabel('Densidade de Kernel')
plt.title('STI = 0.9')
plt.show()
```

12.1.1.1 - MAPEAMENTO DOS DADOS DE ALVO (MMD - Target Domain)

```
z_mapping_VAE_target_domain = dataset
z_mapping_VAE_target_domain
```

10 12.1.2 - KDE 2D Visualization of the bidmensional encoded space and STI values (Target Domain)

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import kde

######## create data
plt.figure(figsize=(6, 6))
is_STI_1 =  dataset['STI'] == 1
is_STI_1_data = dataset[is_STI_1]

x = is_STI_1_data['z1']
y = is_STI_1_data['z2']

# Evaluate a gaussian kde on a regular grid of nbins x nbins over data extents
```

```
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()]))

# Make the plot
plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='auto')

plt.xlabel('Dimensão latente - z1')
plt.ylabel('Dimensão latente - z2')
plt.title('STI = 0.1')
plt.show()
#####
######## create data
plt.figure(figsize=(6, 6))
is_STI_1 =  dataset['STI'] == 3
is_STI_1_data = dataset[is_STI_1]

x = is_STI_1_data['z1']
y = is_STI_1_data['z2']

# Evaluate a gaussian kde on a regular grid of nbins x nbins over data extents
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()]))

# Make the plot
plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='auto')

plt.xlabel('Dimensão latente - z1')
plt.ylabel('Dimensão latente - z2')
plt.title('STI = 0.3')
plt.show()
#####
######## create data
plt.figure(figsize=(6, 6))
is_STI_1 =  dataset['STI'] == 6
is_STI_1_data = dataset[is_STI_1]

x = is_STI_1_data['z1']
y = is_STI_1_data['z2']

# Evaluate a gaussian kde on a regular grid of nbins x nbins over data extents
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()]))

# Make the plot
plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='auto')

plt.xlabel('Dimensão latente - z1')
plt.ylabel('Dimensão latente - z2')
plt.title('STI = 0.6')
plt.show()

######## create data
'''
```

```
plt.figure(figsize=(6, 6))
is_STI_1 =  dataset['STI'] == 9
is_STI_1_data = dataset[is_STI_1]

x = is_STI_1_data['z1']
y = is_STI_1_data['z2']

# Evaluate a gaussian kde on a regular grid of nbins x nbins over data extents
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()]))

# Make the plot
plt.pcolormesh(xi, yi, zi.reshape(xi.shape), shading='auto')

plt.xlabel('Dimensão latente - z1')
plt.ylabel('Dimensão latente - z2')
plt.title('STI = 0.9')
plt.show()
'''
```

12.2 - Cálculo da métrica MMD e visualização

12.2 - MMD metric calculation and visualization of the Maximum Mean Discrepancy (MMD)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
from scipy.stats import dirichlet
from torch.distributions.multivariate_normal import MultivariateNormal

import torch
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

m = 20 # sample size, a smal number for the opimzation

x_mean = torch.FloatTensor(([z_mapping_VAE_source_domain["z1"].mean(),
            z_mapping_VAE_source_domain["z2"].mean()]))

#t orch.zeros(2) + 1 #(pegue as médias z1 e z2 form Source)

y_mean = torch.FloatTensor(([z_mapping_VAE_target_domain["z1"].mean(),
            z_mapping_VAE_target_domain["z2"].mean()]))

#torch.zeros(2)     #(pegue as médias z1 e z2 form Target)

# I can calculate the both
x_cov =  torch.FloatTensor(
                    np.cov([z_mapping_VAE_source_domain["z1"],
                        z_mapping_VAE_source_domain["z2"]])
                        ) * torch.eye(2) #2*torch.eye(2)

# IMPORTANT: Covariance matrices must be positive definite
y_cov = torch.FloatTensor(
                    np.cov([z_mapping_VAE_target_domain["z1"],
                        z_mapping_VAE_target_domain["z2"]])
                        ) * torch.eye(2)
```

```
# Making the sampling
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):
    """Emprical maximum mean discrepancy. The lower the result
       the more evidence that distributions are the same.

    Args:
        x: first sample, distribution P
        y: second sample, distribution Q
        kernel: kernel type such as "multiscale" or "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 result of X and Y is {result.item()}")

# ---- Plotting setup ----

fig, (ax1,ax2) = plt.subplots(1,2,figsize=(8,4), dpi=100)
# plt.tight_layout()
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("Distribution of $X \sim P(X)$")
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="$X$ Samples",
            marker="o",
            facecolor="r",
            edgecolor="k")

ax1.legend()

CS2 = ax2.contourf(x1, x2, qy_grid,100, cmap=plt.cm.YlGnBu)
ax2.set_title("Distribution of $Y \sim Q(Y)$")
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="$Y$ Samples",
            marker="o",
            facecolor="r",
            edgecolor="k")
ax2.legend()

# ax1.axis([-2.5, 2.5, -2.5, 2.5])

# Add colorbar and title
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('Density results')
plt.suptitle(f"MMD result: {round(result.item(),3)}",
             y=0.95,
             fontweight="bold")

plt.show()
```

12.3 - Previsão em lote para o conjunto de dados x_tr

12.3 - Batch Prediction for the x_tr Dataset

```
print(x_tr[0].shape)
x_tr_latent = encoder.predict(x_tr, batch_size=batch_size)

# This result is gonna be the reference value for the Loss function
svr.predict(x_tr_latent)
print("This was the batch prediction")
```

12.4 - Previsão de amostra única no domínio de destino usando o modelo de codificador treinado no domínio de origem

12.4 - Single Sample Prediction in the Target Domain Using the Encoder Model Trained on the Source Domain

```
print(x_tr[0:2].shape)

x_tr_latent = encoder.predict(x_tr[0:2], batch_size=batch_size)
x_tr_latent

svr.predict(x_tr_latent)
```

12.5 - Validação de modelo: validação de dimensão

12.5 - Model Validation: Dimension Validation

```
# No verifies the testing dataset
print("Input sinal bruto - Dimensão bruta antes de formatar:", x_test.shape)

print("Dimensão após restruturação para servir com a input do autoencoder:",
      x_test.reshape(x_test.shape[0],
      img_rows).shape)

x_test_reshaped = x_test.reshape(x_test.shape[0], img_rows)

# configura input do autoencoder (batch )
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

# Lembrar que tenho que fazer a padronização para predição
# (padronização do input max, minx), já configurado.
svr.predict(x_tr_latent_pred)
```

13 - Modelagem de aprendizagem de transferência

13 - Transfer Learning Modeling

```
import numpy as np
import keras
```

```
from keras.layers import Input
from keras.layers.core import Dense, 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
from keras import backend as K
import matplotlib.pyplot as plt
import tensorflow as tf

# verifiyng the shape
print(x_tr.shape)
print(y_train.shape)

from sklearn.model_selection import train_test_split

### 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)

# Balanced dataset
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
over_sampler = RandomOverSampler(random_state=42)

X_res, y_res = over_sampler.fit_resample(X_train, y_train)

print(f"Training target statistics: {Counter(y_res)}")
# print(f"Testing target statistics: {Counter(y_test)}")

X_train, y_train = X_res, y_res/10
```

13.1 - Carga da função de perda KCPR usada na equação MVPanP

13.1 - Loading the KCPR Loss Function Used in the MVPAnP Equation

```
# Load
svr_persited  = load('srv_reg.joblib')
svr.predict(x_tr_latent)

# Load vae
encoder_reconstruted = keras.models.load_model("my_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

svr.predict([[1., 1.0]])
```

13.2 - Implementação da função KCPR Loss (Custom Tensor Function) no modelo Eager

13.2 - Implementation of the KCPR Loss Function (Custom Tensor Function) in Eager Mode

```
# from numpy import linalg as LA
kl = tf.keras.losses.KLDivergence()

# Reconstruting the VAE model trained on the Source Domain
encoder_reconstruted = keras.models.load_model("my_model.model")

def KPCR_Prediction_Loss(input_tensor):

    z_latent = tensorflow_graph(np.expand_dims(input_tensor, axis=0),
                                False, None).numpy()

    src_pred = svr.predict(z_latent)

    return src_pred # np.sinh(input_tensor)

@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 KPCR_loss(input_tensor):

    latent_pred = encoder_reconstruted.predict(
                                        input_tensor.numpy(),
                                        batch_size=1)

    return 0 #svr_persited.predict(latent_pred)
'''

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)

# MVPAnP_Loss --> Loss function sum of components
# MVPAnP_Loss --> custom_loss2
# my_numpy_func --> KPCR_Prediction_Loss
# tf_function(data) # KPCR_loss(data )
```

13.3 - Criando um modelo simples de MLP (Multilayer Perceptron) para testar a função de perda personalizada

13.3 - Creating a Simple MLP (Multilayer Perceptron) Model to Test the Custom Loss Function

```
# create model
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)
#o = Dense(1, kernel_initializer='normal', activation='linear')(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)

#model.fit(X_train, [X_train,y_train], epochs=1, batch_size=1,verbose=1)

type(X_train)

print(X_train.shape)
print(y_train.shape)

data = np.column_stack((y_train, X_train))
data.shape

"""
history = model.fit(X_train,
                    [X_train,y_train],
                    epochs=1,
                    batch_size=1,
                    verbose=0)
"""

# list all data in history
print(history.history.keys())

# summarize history for loss
plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# Evalute model
score = model.evaluate(X_train, np.column_stack((y_train, X_train)), verbose=1)
print('Test loss:', score)

# Make a prediction
yhat = model.predict(X_train)
yhat.shape

y_train
```

13.4 - Validação do modelo MVPanP: usando otimização de hiperparâmetros e ANOVA

13.4 - MVPAnP Model Validation: Using Hyperparameter Optimization and ANOVA

```
"""
Optuna example that optimizes a neural network
classifier configuration for the
```

```
MNIST dataset using Keras.
In this example, we optimize the validation accuracy of
MNIST classification using
Keras. We optimize the filter and kernel size,
kernel stride and layer activation.
# reference:
https://github.com/optuna/optuna-examples/blob/main/keras/keras_simple.py
"""
import urllib
import warnings

import optuna

from keras.backend import clear_session
from keras.datasets import mnist
from keras.layers import Conv2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.models import Sequential
from tensorflow.keras.optimizers import RMSprop
```

13.4.1 - Otimização de Hiperparâmetros para Modelo de Linha de Base (Rede Neural Convolucional 1D) sem Transfer Learning

13.4.1 - Hyperparameter Optimization for Baseline Model (1D Convolutional Neural Network) without Transfer Learning

```
def start_Baseline(features1,features2, trials,  plot_graph = False):

  # 1D CONV NET MODEL
  def create_model(activation,
                   dropout_rate,
                   filters,
                   kernel_size,
                   strides,
                   kernel_initializer):

    """
    Create an temporary model
    """
    clear_session()

    '''
    # 1D CNN neural network - 1D CNN model starts
    model_m.add(Reshape((TIME_PERIODS, num_sensors),
                        input_shape=(input_shape,)))

    model_m.add(Conv1D(100, 10, activation='relu',
                input_shape=(TIME_PERIODS,
                num_sensors)))

    i = Reshape((1600, 1), input_shape=(1600,))

    filters = num_filters[0]
    print("FILTER VALUES ----------" , filters)
    '''

    if type(filters) == tuple:
        filter_num = filters[0]
    else:
```

```
    filter_num = filters

  print("FILTER VALUES ----------" , filters)
  print("FILTER VALUES ----------" , filter_num)

  i = Input(shape=(1600,1))
  #x = Conv1D(100, 10, kernel_initializer=kernel_initializer,
  #activation=activation, input_shape=(1600, 1))(i)
  x = Conv1D(filters=filter_num,
             kernel_size=kernel_size,
             strides=strides,
             kernel_initializer=kernel_initializer,
             activation=activation,
             input_shape=(1600, 1))(i)
  #i = Input(shape=(1600,))
  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 CNN model ends
  '''
  i = Input(shape=(1600,))
  x = Dense(50, kernel_initializer=kernel_initializer,
            activation=activation)(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')
  model.compile(loss='mean_squared_error', optimizer='adam')

  return model

# Objective function to optimize by 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",)

    # Implement early stopping criterion.
    # Training process stops when there is
    # no improvement during 50 iterations
    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)

# Create final model with the best hyperparams
print('Best hyperparams found by 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.fit(X_train, np.column_stack((y_train, X_train)),
# epochs=5, batch_size=1,verbose=0)
# model.compile(optimizer='rmsprop')

model.summary()

# Implement early stopping criterion.
# Training process stops when there is no improvement during 50 iterations
callback = keras.callbacks.EarlyStopping(monitor='loss', patience=50)
history = model.fit(features1,
                    features2,
                    batch_size = 1,
                    epochs=2,
                    callbacks = [callback],
                    verbose = 0)
```

```
  result = model.predict(features1)

  # Result evaluation
  print(f'RMSE Autoencoder: {np.sqrt(mean_squared_error(features2, result))}')
  print('')

  # Following values are returned: extracted_f || MSE || OPTUNA best hyperparams

  return mean_squared_error(features2, result), study.best_params, study

Acoder_MSE, Acoder_hyperparams, Study = start_Baseline(X_train, y_train,
                              trials = 10,
                              plot_graph=True)

# save results
df_results_without_TF = Study.trials_dataframe()

# df_results.to_pickle(results_directory + 'df_optuna_results.pkl')
# df_results.to_csv(results_directory + 'df_optuna_results.csv')
df_results_without_TF

print("Number of finished trials: {}".format(len(Study.trials)))

print("Best trial:")
trial = Study.best_trial

print("  Value: {}".format(trial.value))

print("  Params: ")
for key, value in trial.params.items():
    print("    {}: {}".format(key, value))
```

13.4.2 - Otimização de hiperparâmetros para modelo de linha de base (1D CONVNET) - COM Transfer Learning

13.4.2 - Hyperparameter Optimization for Baseline Model (1D CONVNET) - WITH Transfer Learning

```
def start_Benchmarking(features1,features2, trials,  plot_graph = False):

  # 1D CONV NET MODEL
  def create_model(activation,dropout_rate,
                   filters, kernel_size,
                   strides, kernel_initializer):

    clear_session()
    #from numpy import linalg as LA
    kl = tf.keras.losses.KLDivergence()
    encoder_reconstruted = keras.models.load_model("my_model.model")

    def KPCR_Prediction_Loss(input_tensor):

        z_latent = tensorflow_graph(np.expand_dims(input_tensor, axis=0),
                                    False, None).numpy()

        src_pred = svr.predict(z_latent)

        return src_pred # np.sinh(input_tensor)
```

```
@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 KPCR_loss(input_tensor):

    latent_pred = encoder_reconstruted.predict(input_tensor.numpy(), batch_size=1)

    return 0 #svr_persited.predict(latent_pred)
'''

def MVPAnP_Loss(data, y_pred):
    y_true = data[0][0]
    input2latent = data[0][1:1601]
    # MSE loss function

    return K.mean(K.square(y_pred - y_true), axis=-1) - 0.1*kl(KPCR_loss(input2latent), y_true)

# tf_function(data) # KPCR_loss(data )
# create model

tf.config.run_functions_eagerly(True)

# Bug fix
if type(filters) == tuple:
    filter_num = filters[0]
else:
    filter_num = filters

print("FILTER VALUES ----------" , filters)
print("FILTER VALUES ----------" , filter_num)

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

# Objective function to optimize by 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")

  # Implement early stopping criterion.
  # Training process stops when there is no improvement during 50 iterations

  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)

# Create final model with the best hyperparams
print('Best hyperparams found by 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()
# Implement early stopping criterion.
# Training process stops when there is no improvement during 50 iterations
callback = keras.callbacks.EarlyStopping(monitor='loss', patience=50)
history = model.fit(features1,
                    features2,
```

```
                batch_size = 1,
                epochs=2,
                callbacks = [callback],
                verbose = 0)

  result = model.predict(features1)

  # Result evaluation
  print(f'RMSE Autoencoder: {np.sqrt(mean_squared_error(features2[:,0], result))}')

  # Following values are returned: extracted_f || MSE || OPTUNA best hyperparams

  return mean_squared_error(features2[:,0], result), study.best_params, study

Acoder_MSE, Acoder_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()
#df_results.to_pickle(results_directory + 'df_optuna_results.pkl')
#df_results.to_csv(results_directory + 'df_optuna_results.csv')
df_results_with_TF

df_results_with_TF["value"].values

print("Number of finished trials: {}".format(len(Study.trials)))

print("Best trial:")
trial = Study.best_trial

print("  Value: {}".format(trial.value))

print("  Params: ")
for key, value in trial.params.items():
    print("    {}: {}".format(key, value))
```

13.5 - Análise ANOVA para o modelo de linha de base antes e depois do TF

13.5 - ANOVA analysis for the baseline model before and after TF

```
import pandas as pd
from scipy import stats
from statsmodels.graphics.gofplots import qqplot
from matplotlib import pyplot as plt
import numpy as np
from scipy.stats import f_oneway
from statsmodels.stats.multicomp import pairwise_tukeyhsd
print('Libraries imported')

df_results_with_TF["Transfer_Learning"] = "Com TF"
df_results_without_TF["Transfer_Learning"] = "Sem TF"

# Merge pandsa dataframe
benchmarking_analysis = pd.concat([df_results_without_TF, df_results_with_TF])

# Formated the decimal places
benchmarking_analysis['value'] = benchmarking_analysis['value'].values.round(4)
```

```
benchmarking_analysis['value'].values.round(4)
```

14 - Comparação dos resultados dos métodos propostos de TF

14 - Comparison of the results of the proposed methods of TF

```
benchmarking_analysis

xx = benchmarking_analysis[benchmarking_analysis['Transfer_Learning'] == 'Sem TF']['value'].values
xy = benchmarking_analysis[benchmarking_analysis['Transfer_Learning'] == 'Com TF']['value'].values


# Importing library
from scipy.stats import f_oneway
# Conduct the one-way ANOVA
f_oneway(xx, xy)
```

14.1 - Análise ANOVA e boxplot

14.1 - Anova Analysis and BoxPlot

```
# Import libraries
import matplotlib.pyplot as plt
import numpy as np


# Creating dataset
np.random.seed(10)

data = [xx, xy]
plt.rcParams['font.size'] = '16'
fig = plt.figure(figsize =(10, 5))

# Creating axes instance
ax = fig.add_axes([0, 0, 1, 1])

# Creating plot
bp = ax.boxplot(data)

# x-axis labels
ax.set_xticklabels(['Sem Transferência', 'Com transferência'])
#ax.set_yticklabels("Means Square erro")
ax.set_ylabel("Means Square Error")

# show plot
plt.show()

from statsmodels.stats.multicomp
import pairwise_tukeyhsd

tukey = pairwise_tukeyhsd(endog=benchmarking_analysis['value'], groups=benchmarking_analysis ['T
ransfer_Learning'], alpha=0.05)
print(tukey)
```