Vector-Quantized Variational Autoencoder para Audio Denoising com U-Net e SE blocks

1st Diana Rocha
Centro de Informática - UFPE
Recife, Brazil
drs4@cin.ufpe.br

2nd Filip Fernandes
Centro de Informática - UFPE
Recife, Brazil
fapf@cin.ufpe.br

***Abstract*—Neste estudo, propomos um modelo inovador de audio denoising utilizando um híbrido de Vector Quantized Variational Autoencoder (VQVAE) e U-Net combinado com a Transformada de Fourier de Tempo Curto (STFT) para processamento de espectrogramas limpos e ruidosos. O VQVAE é treinado para aprender representações discretas das distribuições latentes dos sinais de fala limpos, permitindo a reconstrução de fala a partir de entradas corrompidas por ruído. A arquitetura U-Net com SE blocks aprimora a representatividade da rede e a capacidade de lidar com ruídos complexos, enquanto o processamento STFT facilita a análise e manipulação do sinal em diferentes faixas de frequência. A combinação do VQVAE com a arquitetura U-Net, SE blocks e STFT oferece uma solução eficaz para melhorar a comunicação humana e a experiência auditiva em ambientes com ruído de fundo.**

I. Introdução

A qualidade do áudio é essencial para uma comunicação eficaz e uma experiência auditiva agradável. No entanto, a proliferação de sistemas de som de baixa qualidade compromete essa experiência, especialmente em dispositivos móveis e ambientes públicos. Métodos tradicionais de separação de ruído, como a subtração espectral, apresentam limitações significativas, introduzindo distorções em cenários complexos.

Uma abordagem promissora para melhorar a qualidade do áudio é o uso de autoencoders variacionais com quantizador vetorial (VQ-VAE), que aprendem representações latentes dos espectrogramas de voz. Para aprimorar ainda mais a capacidade de reconstrução e separação de áudio, incorporamos também arquiteturas avançadas como a U-Net e blocos de squeeze-and-excitation (SE). A U-Net, com seu contracting path e expanding path, permite a recuperação detalhada de características espaciais, enquanto os SE blocks ajustam dinamicamente a importância de diferentes canais de informação. Combinando estas técnicas, a abordagem proposta do híbrido de VQ-VAE e UNet visa garantir uma fidelidade espectral superior e uma percepção auditiva mais agradável, resultando em uma separação de voz mais clara e precisa.

II. Modelos de Variáveis Latentes

A. Definição e Fundamentação

Modelos de variável latente são modelos probabilísticos onde as variáveis observáveis $\mathbf{x}$ são explicadas por variáveis latentes não observáveis $\mathbf{z}$. As variáveis latentes capturam a estrutura subjacente dos dados e permitem uma representação mais compacta.

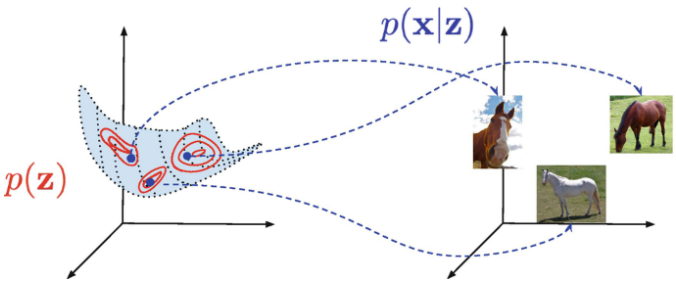


Fig. 1: Variáveis Latentes e Dados Observáveis

B. Modelagem Probabilística

1) Distribuição Priori das Variáveis Latentes:

$$\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{z}) \tag{1}$$

Onde $p_{\boldsymbol{\theta}}(\mathbf{z})$ é a distribuição a priori, parametrizada por $\boldsymbol{\theta}$, que modela o comportamento das variáveis latentes $\mathbf{z}$, geralmente assumida como uma distribuição gaussiana.

2) Distribuição Condicional das Variáveis Visíveis:

$$\mathbf{x}|\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \tag{2}$$

Onde $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ é a distribuição condicional das variáveis visíveis $\mathbf{x}$ dada $\mathbf{z}$, também parametrizada por $\boldsymbol{\theta}$. Define como as variáveis latentes são mapeadas de volta para os dados observáveis.

3) Inferência:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p(\mathbf{x})} \tag{3}$$

Onde $p(\mathbf{z}|\mathbf{x})$ é a distribuição a posteriori das variáveis latentes dado os dados observáveis. Utilizamos a regra de Bayes para calcular esta distribuição.

C. Geração

A geração de novos dados implica em amostrar $\mathbf{z}$ da distribuição a priori $p_{\boldsymbol{\theta}}(\mathbf{z})$ e então amostrar $\mathbf{x}$ de $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$:

$$\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{z}), \quad \mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \tag{4}$$

Por outro lado, a inferência das variáveis latentes envolve amostrar $\mathbf{x}$ da distribuição marginal $p(\mathbf{x})$ e, em seguida, calcular $\mathbf{z}$ a partir da distribuição a posteriori $p(\mathbf{z}|\mathbf{x})$:

$$\mathbf{x} \sim p(\mathbf{x}), \quad \mathbf{z} \sim p(\mathbf{z}|\mathbf{x}) \tag{5}$$

D. Problema da Verossimilhança Marginalizada

A verossimilhança marginalizada $p(\mathbf{x})$ é dada por:

$$p(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z})\, d\mathbf{z} \tag{6}$$

Esta integral é geralmente intratável devido à alta dimensionalidade e à complexidade das distribuições envolvidas.

III. Inferência Variacional

Para superar a intractabilidade da verossimilhança marginalizada, recorremos à inferência variacional. Introduzimos uma nova distribuição, denominada $q_{\boldsymbol{\psi}}(\mathbf{z})$, que tenta aproximar $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. Essa nova distribuição será obtida por meio de um problema de otimização com respeito aos parâmetros $\boldsymbol{\psi}$.

A. O objetivo variacional

A inferência variacional busca minimizar a divergência de Kullback-Leibler (KL), também chamada de *variational gap*, entre $q_{\boldsymbol{\psi}}(\mathbf{z})$ e $p(\mathbf{z}|\mathbf{x})$, otimizando $\boldsymbol{\psi}$. Para um único $\mathbf{x}$, o objetivo fica definido como:

$$\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} D_{KL}(q_{\boldsymbol{\psi}}(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x})) \tag{7}$$

Expandindo a divergência KL obtemos

$$\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} \mathbb{E}_{q_{\boldsymbol{\psi}}(\mathbf{z})}[\log q_{\boldsymbol{\psi}}(\mathbf{z}) - \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - \log p_{\boldsymbol{\theta}}(\mathbf{z})] \tag{8}$$

O termo aditivo $\log p_\theta(\mathbf{x})$ foi omitido, pois é constante com respeito a $\boldsymbol{\psi}$. Assim, o problema de otimização se torna maximizar o *evidence lower bound* (ELBO) através de $\boldsymbol{\psi}$

$$Ł(\boldsymbol{\theta}, \boldsymbol{\psi}|\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\psi}}(\mathbf{z})}[\log q_{\boldsymbol{\psi}}(\mathbf{z}) - \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - \log p_{\boldsymbol{\theta}}(\mathbf{z})] \tag{9}$$

Uma outra forma de ver o ELBO, que será útil no entendimento dos VAEs é a seguinte

$$Ł(\boldsymbol{\theta}, \boldsymbol{\psi}|\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\psi}}(\mathbf{z})}[p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - D_{KL}(q_{\boldsymbol{\psi}}(\mathbf{z}) || p_{\boldsymbol{\theta}}(\mathbf{z}))] \tag{10}$$

Assim, vemos que o objetivo é o *maximum likelihood estimate* (MLE) mais um termo de regularização que penaliza a posterior variacional caso ela se torne diferente da priori.

B. Estimando $\boldsymbol{\theta}$

Supondo que nosso modelo seja da forma

$$p(\mathcal{D}|\mathbf{z}_{1:N}) = \prod_{n=1}^{N} p(\mathbf{z}_n|\boldsymbol{\theta}) p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\theta}) \tag{11}$$

Se quisessemos computar o MLE para $\boldsymbol{\theta}$, seria preciso marginalizar com respeito às variáveis latentes, assim voltaríamos a estar tentando calcular a mesma marginal intratável. Felizmente, o ELBO é uma margem inferior no MLE

$$Ł(\boldsymbol{\theta}, \boldsymbol{\psi}_n|\mathbf{x}_n) \leq \log p_{\boldsymbol{\theta}}(\mathbf{x}_n) \tag{12}$$

Assim, apenas com o ELBO, conseguimos otimizar $\boldsymbol{\psi}$ e $\boldsymbol{\theta}$ para todo o *dataset* dessa forma

$$Ł(\boldsymbol{\theta}, \boldsymbol{\psi}_{1:N}|\mathcal{D}) = \sum_{n=1}^{N} Ł(\boldsymbol{\theta}, \boldsymbol{\psi}_n|\mathbf{x}_n) \leq \log p(\mathcal{D}|\boldsymbol{\theta}) \tag{13}$$

C. Inferência variacional amortizada

Como é necessário encontrar $\boldsymbol{\psi}_n$ para cada $\mathbf{x}_n$, uma alternativa é treinar um novo modelo, chamado de rede de inferência, para prever $\boldsymbol{\psi}_n$ dado $\mathbf{x}_n$, usando $\boldsymbol{\psi}_n = f_\phi^{\text{inf}}(\mathbf{x}_n)$. Para sermos breves, escreveremos o posterior amortizado dessa forma

$$q(\mathbf{z}_n|\mathbf{x}_n) = q(\mathbf{z}_n|f_\phi^{\text{inf}}(\mathbf{x}_n)) = q_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{x}_n) \tag{14}$$

IV. Variational Autoencoders (VAE)

Os Variational Autoencoders são, portanto, modelos de variável latente que utilizam inferência amortizada. Particularmente, a posterior variacional terá os parâmetros estimados pela rede inferencial, o *encoder*, enquanto que o *decoder* será uma rede neural parametrizada por $\boldsymbol{\theta}$ que estimará os parâmetros de $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Consideraremos que $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ e $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ pertencem à família de distribuições exponenciais, como Gaussianas ou produtos de Bernoullis, e que a priori $p_{\boldsymbol{\theta}}(\mathbf{z})$ é uma distribuição Gaussiana com parâmetros fixos.

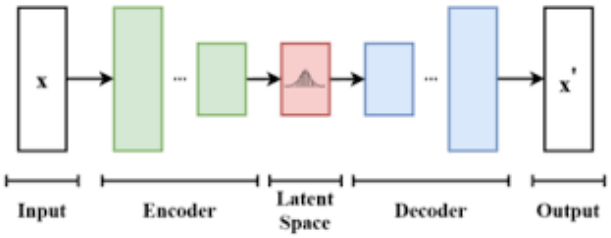


Fig. 2: Estrutura do VAE

A. Estrutura do VAE

Os VAEs possuem os mesmos componentes que modelo de variável latente explicado na introdução. Assim combinam a inferência variacional com autoencoders, consistindo de um encoder e um decoder.

1) Encoder:

$$\mathbf{z}|\mathbf{x} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \tag{15}$$

O encoder mapeia a entrada $\mathbf{x}$ para a variável latente $\mathbf{z}$ através de uma rede neural parametrizada por $\boldsymbol{\phi}$. No espaço latente, assumimos uma distribuição à priori $\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{z})$, que será utilizada após o treino para gerar dados.

2) Decoder:

$$\mathbf{x}|\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \tag{16}$$

O decoder reconstrói a entrada $\mathbf{x}$ a partir da variável latente $\mathbf{z}$ através de uma rede neural parametrizada por $\boldsymbol{\theta}$.

B. Treinamento

Durante o treinamento, objetivo do VAE é minimizar a divergência KL entre $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ e $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. Essa minimização é realizada maximizando a ELBO:

$$Ł(\boldsymbol{\theta}, \boldsymbol{\psi}|\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\psi}}(\mathbf{z})}[p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - D_{KL}(q_{\boldsymbol{\psi}}(\mathbf{z})||p_{\boldsymbol{\theta}}(\mathbf{z}))] \quad (17)$$

Na prática, se utilzarmos Gaussianas, a verossimilhança pode ser otimizada com erro quadrático médio, e a divergência KL também pode ser otimizada de maneira simples, pois no caso das Gaussianas, possui forma fechada.

C. Inferência e Geração

Durante a inferência, é possível obter uma representação latente da entrada através do *encoder*. Para a geração, o VAE gera novos dados através da amostragem da distribuição a priori, seguida de seu uso como input para o *decoder*. Se objetivo for reconstruir um dado de entrada, como é comum no *denoising*, é possível utilzar o VAE como um *Autoencoder* convencional, bastando, no lugar de fazer uma mostragem, obter a média da distribuição latente, fazendo assim uma estimação pontual que representa o que o *encoder* acredita.

V. Vector-Quantized VAE (VQ-VAE)

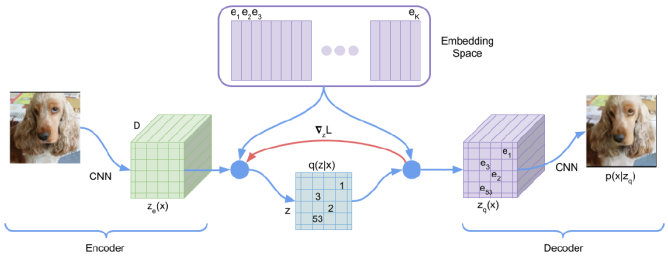


Fig. 3: O modelo VQ-VAE

O Vector-Quantized VAE é um caso particular do VAE. Em especial, utiliza um espaço latente discreto, o que permite capturar representações mais nuançadas e estruturadas, o que é particularmente vantajoso para tarefas como redução de ruído em áudio.

Consideraremos a distribuição a priori sendo uniforme. Assim, o termo da divergência Kullback-Leibler se reduz a uma constante, podendo ser desconsiderada da otimização.

No caso de imagens, podemos obter um modelo expressivo usando um tensor 3D de latentes discretos, $z \in \mathbb{R}^{H \times W \times K}$, onde K é o número de valores discretos por variável latente. Nós o comparamos a um *codebook* de vetores de *embedding*, $\{e_k : k = 1 : K, e_k \in \mathbb{R}^L\}$, e então definimos z_{ij} como o índice da entrada do *codebook* mais próxima.

$$q(z_{ij} = k \mid x) = \begin{cases} 1 & \text{se } k = \arg\min_{k'} \|z_e(x)_{i,j,:} - e_{k'}\|_2 \\ 0 & \text{caso o contrário} \end{cases}$$

Quando reconstruímos a entrada, substituímos cada índice de código discreto pelo vetor de código correspondente com valor real:

$$(z_q)_{ij} = \mathbf{e}_k \text{ onde } z_{ij} = k \quad (18)$$

Esses valores são então passados para o *decoder*, $p(x|z_q)$, como no VAE tradicional. Note que, embora z_q seja gerado a partir de uma combinação discreta de vetores de código, o uso de um código discretizado torna o modelo muito expressivo. Por exemplo, se usarmos uma grade de 32×32, com $K = 512$, então podemos gerar $512^{32 \times 32} = 2^{9,216}$ imagens distintas, o que é astronomicamente grande.

Para ajustar este modelo, podemos minimizar a log-verossimilhança negativa (erro de reconstrução) usando o estimador straight-through. Isso equivale a passar os gradientes da entrada do *encoder* $z_q(x)$ para a saída do *encoder* $z_e(x)$. Infelizmente, isso significa que as entradas do *codebook* não receberão nenhum sinal de aprendizado.

Para resolver isso, os autores propuseram adicionar um termo extra à perda, conhecido como *codebook loss*, que incentiva as entradas do código e a corresponder à saída do *encoder*. Tratamos o *encoder* $z_e(x)$ como um alvo fixo, adicionando um operador de parada de gradiente a ele; isso garante que z_e seja tratado normalmente na passagem direta, mas tenha gradiente zero na passagem reversa. A perda modificada (eliminando os índices espaciais i, j) torna-se

$$\mathcal{L} = -\log p(x|z_q(x)) + \|\text{sg}(z_e(x)) - e\|_2^2 \quad (19)$$

onde e se refere ao vetor do *codebook* atribuído a $z_e(x)$, e sg é o operador *stop gradient*.

Entretanto, também é importante garantir que o *encoder* não "mude de ideia" com muita frequência sobre qual valor do *codebook* usar. Para evitar isso, os autores propõem adicionar um terceiro termo à perda, conhecido como *commitment loss*, que incentiva a saída do *encoder* a estar próxima aos valores do *codebook*. Assim, obtemos a perda final:

$$\mathcal{L} = -\log p(x|z_q(x)) + \|sg(z_e(x)) - e\|_2^2 + \beta\|z_e(x) - sg(e)\|_2^2 \quad (20)$$

VI. Dataset

O conjunto de dados utilizado para o treinamento e avaliação do modelo é o *PolyAI/minds14 en-US*. Este dataset é composto por aproximadamente 1 hora e 15 minutos de áudios em inglês, no formato de vetores. A seguir, detalharemos o pré-processamento necessário para adequar os dados ao formato adequado para o treinamento.

VII. Pré-processamento

O pré-processamento dos dados de áudio é um passo crucial para preparar os dados para o treinamento do modelo de denoising. Esta seção descreve o processo de divisão dos vetores de áudio em trechos de um segundo, a transformação dos trechos em espectrogramas utilizando a Transformada de Fourier de Tempo Curto (STFT), e a padronização dos espectrogramas para melhorar a eficiência do treinamento.

A. Divisão dos Vetores de Áudio em Trechos de Um Segundo

Primeiramente, os vetores de áudio são divididos em trechos de um segundo. Esta divisão é importante para criar segmentos de áudio uniformes que podem ser processados de maneira consistente pelo modelo. Cada trecho de um segundo assume uma taxa de amostragem de 8000 Hz.

- **Entrada:** Vetores de áudio de comprimento variável.
- **Saída:** Lista de trechos de áudio de um segundo.

B. Transformada de Fourier de Tempo Curto (STFT)

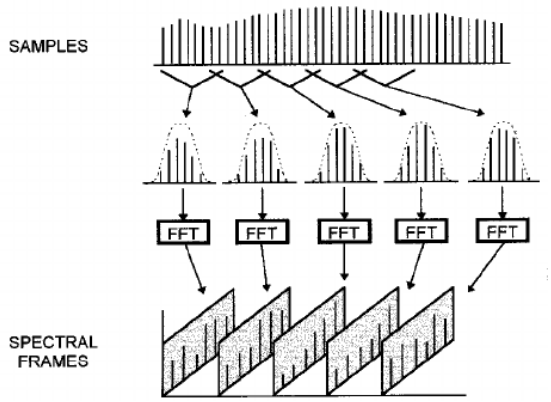
A STFT é uma técnica para transformar um sinal de áudio em uma representação de frequência-tempo. Isso é feito aplicando a Transformada de Fourier a janelas sobrepostas do sinal de áudio. Para este projeto, utilizamos a função STFT do *SciPy*, configurada com os seguintes parâmetros:

- **Frequência de amostragem (fs):** 8000 Hz.
- **Número de pontos por segmento (nperseg):** 519.
- **Número de pontos de sobreposição (noverlap):** 488.

Esses parâmetros foram escolhidos para garantir que os espectrogramas resultantes sejam quadrados, o que é ideal para as operações de convolução realizadas pela arquitetura U-Net.

- **Entrada:** Trechos de áudio de um segundo.
- **Saída:** Espectrogramas de frequência-tempo.

1) Funcionamento da STFT: A STFT divide o sinal de áudio em janelas de tempo de tamanho especificado (nperseg), e cada janela é transformada para o domínio da frequência usando a Transformada de Fourier. As janelas são sobrepostas (noverlap) para capturar melhor as transições no sinal de áudio. O resultado é um espectrograma que representa como a energia do sinal está distribuída ao longo do tempo e das frequências.



(a)

Fig. 4: Short-Time Fourier Transform

C. Padronização dos Espectrogramas

A padronização dos espectrogramas é um passo importante para normalizar os dados, melhorando a eficiência do treinamento do modelo. A padronização transforma os espectrogramas para terem média zero e desvio padrão igual a um.

- **Entrada:** Espectrogramas de frequência-tempo.
- **Saída:** Espectrogramas padronizados, média e desvio padrão.

A padronização ajuda a estabilizar o processo de treinamento, permitindo que o modelo converja mais rapidamente e evitando que certos parâmetros dominem devido a diferentes escalas nos dados.

VIII. Arquitetura do Modelo

A arquitetura do modelo VQVAE-UNet combina técnicas de quantização vetorial e redes neurais convolucionais para alcançar uma representação discreta e detalhada dos dados de entrada. A estrutura base é fundamentada na arquitetura da U-Net, mas incluindo um bloco de Squeeze-and-Excitation em cada um dos seus estágios e um quantizador vetorial em seu bottleneck.

A abordagem da UNet se justifica pela sua grande capacidade de realizar reconstruções complexas, sendo originalmente desenvolvida para segmentação de imagens médicas. Já os SE-blocks foram escolhidos para realçar a representatividade dos feature maps ao recalibrar dinamicamente os canais, numa forma de channel-wise self-attention, amplificando consideravelmente a capacidade do modelo a um custo computacional extra praticamente desprezível.

O modelo implementado se distingue da U-Net convencional ao passar o espaço latente, ou seja, a saída do último layer do encoder, para um quantizador vetorial, onde ele é codificado com o codebook e discretizado, gerando como output os vetores discretos representantes e a quantization loss, exatamente como nos VQ-VAEs. Essas representações discretas, juntamente com as saídas intermediárias dos layers correspondentes do encoder, são passadas para o decoder, onde o espectrograma limpo é progressivamente reconstruído.

A seguir, detalhamos os componentes principais do modelo, incluindo VQVAE, UNet, e Squeeze-and-Excitation Blocks.

A. VQVAE (Vector Quantizer Variational Autoencoder)

O VQVAE é o núcleo do modelo, sendo responsável por quantizar as representações latentes em um espaço discreto. Este processo permite capturar representações mais estruturadas e relevantes dos dados de entrada. O VQVAE é composto por três componentes principais: o Codificador, o Quantizador Vetorial, e o Decodificador.

1) Codificador (Encoder): O Codificador transforma os dados de entrada em representações latentes de alta dimensão. Esta etapa envolve a aplicação de várias camadas convolucionais seguidas por operações de pooling. As camadas convolucionais extraem características relevantes dos dados de entrada, enquanto o pooling reduz a dimensionalidade, facilitando o processamento subsequente. Cada camada convolucional é seguida por uma operação de ativação e normalização para garantir uma melhor convergência durante o treinamento.

2) Quantizador Vetorial (Vector Quantizer): O Quantizador Vetorial mapeia as representações latentes contínuas para um conjunto finito de códigos discretos, utilizando um dicionário

(*codebook*) que é aprendido durante o treinamento. Este processo envolve calcular a distância entre as representações latentes e os vetores do *codebook*, atribuindo cada representação ao vetor mais próximo. A função de perda do VQVAE inclui um termo de compromisso que força as representações latentes a se aproximarem dos vetores do *codebook*, garantindo a discrição das representações.

3) Decodificador (Decoder): O Decodificador reconstrói os dados de entrada a partir dos códigos latentes quantizados. Ele usa camadas convolucionais transpostas para aumentar a dimensionalidade das representações latentes, revertendo o processo de redução do codificador. O Decodificador também aplica operações de ativação e normalização para refinar a reconstrução, assegurando que os dados de saída preservem as características essenciais dos dados de entrada.

B. UNet

O UNet é uma arquitetura de rede neural convolucional que se destaca em tarefas de segmentação e reconstrução de imagens. No contexto do VQVAE-UNet, o UNet é utilizado para processar espectrogramas de áudio, fornecendo uma reconstrução detalhada e preservando a estrutura dos dados.

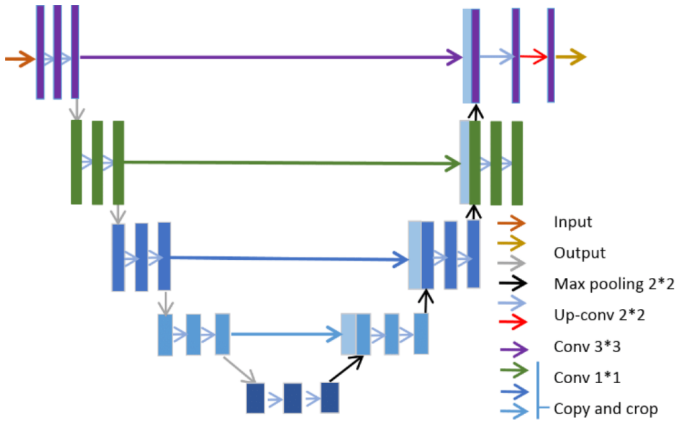


Fig. 5: Arquitetura U-Net

1) Estrutura do UNet: A arquitetura do UNet é composta por dois caminhos principais: um caminho de codificação (downsampling) e um caminho de decodificação (upsampling). O caminho de codificação é responsável por reduzir a dimensionalidade dos dados de entrada enquanto extrai características relevantes, similar ao codificador do VQVAE. Já o caminho de decodificação aumenta a dimensionalidade das representações latentes, permitindo a reconstrução dos dados de entrada.

2) Conexões de Salto (Skip Connections): As conexões de salto são uma característica crucial da UNet. Elas conectam camadas correspondentes dos caminhos de codificação e decodificação, permitindo que a informação de alta resolução dos estágios iniciais do codificador seja diretamente utilizada no decodificador. Isso ajuda a preservar detalhes finos e melhora a precisão da reconstrução, especialmente em tarefas que exigem alta fidelidade nos dados de saída.

C. SE Block (Squeeze-and-Excitation Block)

Os SE Blocks são utilizados para melhorar a capacidade da rede neural de modelar interdependências entre canais. Eles consistem em duas operações principais: squeeze e excitation.

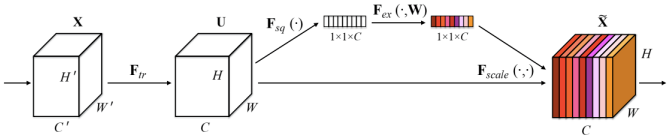


Fig. 6: Arquitetura SE-Block

1) Squeeze: A operação de squeeze calcula a média global de cada mapa de características, reduzindo a dimensionalidade espacial para uma única representação por canal. Isso captura a importância global de cada canal no contexto de toda a imagem ou sinal de entrada.

2) Excitation: A operação de excitation utiliza uma rede neural totalmente conectada para recalibrar a importância de cada canal, aplicando pesos aprendidos às representações dos canais. Este processo permite que a rede ajuste dinamicamente a resposta dos filtros convolucionais de acordo com a importância de cada canal, melhorando a eficiência da extração de características e a precisão da reconstrução.

IX. Treinamento do Modelo

O treinamento do modelo VQVAE-UNet envolve a configuração adequada dos hiperparâmetros, a definição das funções de perda e a otimização do modelo. O processo é cuidadosamente projetado para maximizar a eficiência e a precisão da reconstrução dos dados de áudio. A seguir, detalhamos as etapas e componentes principais do treinamento.

A. Otimizador AdamW

Para a otimização do modelo, utilizamos o otimizador AdamW, uma variante do algoritmo Adam que incorpora regularização de decaimento de peso (*weight decay*) de forma separada. Isso corrige uma limitação do Adam original, onde o decaimento de peso era combinado com a taxa de aprendizado, o que podia levar a problemas de generalização. AdamW melhora a convergência e a capacidade de generalização do modelo.

B. Critérios de Perda

Para treinar o modelo de denoising, além da perda de quantização do Vector Quantizer, utilizamos duas funções de perda principais: MSELoss e L1Loss.

- **MSELoss (Mean Squared Error Loss)**: Esta função mede a média dos quadrados das diferenças entre os valores previstos e os valores reais. É uma medida comum de erro em problemas de regressão.
- **L1Loss (Mean Absolute Error Loss)**: Esta função calcula a média das diferenças absolutas entre os valores previstos e os valores reais. É menos sensível a outliers em comparação com a MSELoss.

A perda total de reconstrução é uma combinação ponderada dessas duas perdas, permitindo um balanceamento entre a suavidade (via MSE) e a robustez (via L1) da reconstrução.

C. Scheduler para Ajuste da Taxa de Aprendizado

Utilizamos um scheduler de taxa de aprendizado chamado *ReduceLROnPlateau*. Este scheduler reduz a taxa de aprendizado quando a métrica monitorada (neste caso, a perda de validação) não melhora após uma quantidade definida de épocas. Isso ajuda a evitar a estagnação do treinamento e melhora a convergência do modelo. O parâmetro *patience* define quantas épocas o scheduler espera antes de reduzir a taxa de aprendizado, enquanto o *factor* define a proporção da redução.

D. Adição de Ruído

Para treinar o modelo de denoising, adicionamos ruído gaussiano aos espectrogramas de entrada. Isso simula as condições de ruído de fundo que o modelo deve aprender a remover. A intensidade do ruído pode ser ajustada conforme necessário para equilibrar a dificuldade da tarefa de denoising.

E. Processo de Treinamento

O processo de treinamento é iterativo, com o modelo alternando entre fases de treinamento e validação:

- Durante a fase de treinamento, o modelo recebe espectrogramas ruidosos, passa-os pelo VQ-VAE e calcula a perda de reconstrução em relação aos espectrogramas originais (sem ruído). A perda total inclui a perda de quantização, que força os códigos latentes a permanecerem próximos dos centros de quantização.
- O otimizador AdamW atualiza os pesos do modelo com base no gradiente da perda total.
- Durante a fase de validação, avaliamos o modelo em um conjunto de dados de teste. Isso ajuda a monitorar o desempenho do modelo e ajustar a taxa de aprendizado através do scheduler.

F. Cálculo e Monitoramento das Perdas

A perda total é calculada como a soma ponderada da MSELoss, L1Loss e perda de quantização. Durante cada época, monitoramos a perda média de treinamento e validação para avaliar a performance do modelo e ajustar hiperparâmetros conforme necessário.

G. Reconstrução de Espectrogramas

Após o treinamento, utilizamos o modelo treinado para reconstruir espectrogramas limpos a partir de espectrogramas ruidosos. A reconstrução é então ”despadronizada” para retornar à escala original dos dados.

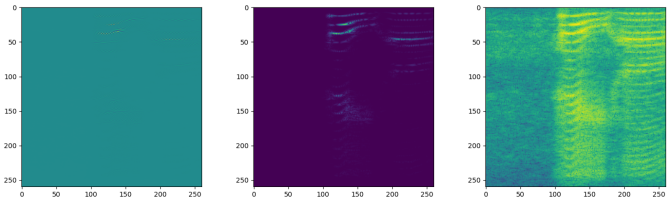
X. Pós-processamento

A. Visualização dos Espectrogramas com Transformação Logarítmica

Os espectrogramas representam a densidade espectral do sinal de áudio ao longo do tempo. No entanto, as magnitudes das frequências podem variar em várias ordens de magnitude, o que dificulta a visualização e a interpretação das informações contidas nos espectrogramas. Para mitigar esse problema, aplicamos a transformação logarítmica na magnitude dos espectrogramas.

- **Realce de Detalhes:** A transformação logarítmica compressa a faixa dinâmica das magnitudes, permitindo que detalhes em frequências de baixa energia sejam mais visíveis.
- **Percepção Humana:** A percepção humana de intensidade sonora é aproximadamente logarítmica, então a visualização logarítmica se alinha melhor com a forma como ouvimos o som.

A transformação é realizada aplicando o logaritmo natural na magnitude do espectrograma, após calcular o valor absoluto para garantir que todas as entradas sejam não negativas.



(a) (b) (c)

Fig. 7: Espectrograma original (a), espectrograma em magnitude (b), espectrograma em log-magnitude (c).

1) Interpretação das Visualizações:

- **Espectrograma Original:** Esta visualização mostra a densidade espectral do sinal de áudio em sua forma original. Devido à ampla faixa dinâmica, pode ser difícil observar detalhes em frequências de baixa energia.
- **Espectrograma com Log-Magnitude:** Esta visualização aplica a transformação logarítmica na magnitude absoluta do espectrograma. Detalhes em frequências de baixa energia tornam-se mais visíveis, facilitando a análise visual da qualidade do denoising.

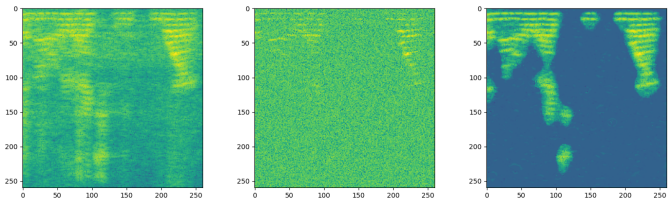
B. Transformada de Fourier Inversa de Tempo Curto (ISTFT)

A ISTFT é o processo inverso da STFT. Utilizamos a função ISTFT do *SciPy* para converter os espectrogramas reconstruídos de volta ao domínio do tempo. Este passo é essencial para avaliar a qualidade do denoising no contexto do sinal de áudio original.

1) Funcionamento da ISTFT: A ISTFT funciona agregando as janelas de tempo que foram transformadas pela STFT. Cada janela de frequência é transformada de volta para o domínio do tempo usando a Transformada de Fourier Inversa. As janelas são então sobrepostas e combinadas para formar o sinal de

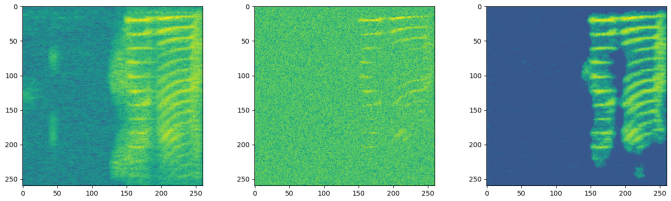
áudio contínuo. Este processo é eficaz para reconstruir sinais que foram processados no domínio da frequência, como no caso de denoising de espectrogramas.

XI. Visualização dos Resultados



(a) (b) (c)

Fig. 8: Comparação entre espectrogramas limpo (a), com ruído (b) e reconstruído sem ruído (c).



(a) (b) (c)

Fig. 9: Exemplo dois

Como pode ser observado nos dois exemplos das figuras 7 e 8, o modelo proposto apresenta uma capacidade de denoising considerável, eliminando o ruído quase que em sua totalidade, e, surprendentemente, não apenas o ruído gaussiano artificialmente adicionado, como também o ruído natural dos espectrogramas antes do pré-processamento, ou seja, teoricamente limpos. Naturalmente, questões sobre eventuais perdas de informação resultantes de um denoising evidentemente tão agressivo foram levantadas, e, diante disso, analisaremos métricas objetivas de avaliação tanto de presença de ruído quanto de qualidade e inteligibilidade sonora, de fato.

XII. Avaliação

O desempenho do modelo foi avaliado por meio das seguintes métricas objetivas:

- STOI (Short-Time Objective Intelligibility): avalia a inteligibilidade da fala aprimorada.
- PESQ (Perceptual Evaluation of Speech Quality): avalia a qualidade perceptual da fala aprimorada.
- NSR (Noise-to-Signal Ratio): mede a relação entre a potência do ruído e a potência do sinal de voz.

TABLE I: Métricas

Audios	Ruidoso vs Original	Denoised vs Original
STOI	0.248	0.852
PESQ	1.178	2.477
NSR	3.520	0.052

Comparando a qualidade, a inteligibilidade e a presença de ruído entre os áudios ruidosos e reconstruídos (denoised) tendo os áudios originais como referência, as métricas objetivas mostraram avanços nas três categorias, com diferenças consideráveis. Para fins de comparação, o STOI varia numa range de 0 a 1, sendo 0 completamente ininteligível e 1 perfeitamente inteligível. Tendo como padrão os áudios originais, os respectivos áudios ruidosos são significativamente menos compreensíveis, com STOI de 0.248, situação que muda com o processo de denoising, que aumenta o STOI para 0.852, consideravelmente elevando a inteligibilidade, embora ainda haja uma diferença em relação ao áudio limpo. De forma semelhante, o modelo também melhorou a qualidade perceptual do áudio em relação à sua versão ruidosa, levando o PESQ de 1.178 para 2.477. Um ponto a se considerar, no entanto, é que a range dessa métrica varia de -0.5 a 4.5, ou seja, apesar do avanço na qualidade comparado ao ruidoso, o áudio reconstruído ainda está longe do que seria considerado ótimo. Já quanto ao NSR, que mede a proporção entre ruído e sinal, uma queda de 3.52 dB para 0.052 dB representa um avanço evidente, indicando uma eliminação substancial de ruído e comprovando objetivamente a eficácia do modelo proposto.

XIII. Conclusão

A abordagem híbrida do VQVAE com a UNet combinada com SE Blocks resulta em um modelo poderoso e eficiente para tarefas de redução de ruído em áudio. Cada componente contribui para a captura de representações estruturadas e detalhadas, garantindo uma reconstrução precisa e de alta fidelidade dos dados de entrada. Apesar de ser inegável a robustez e a capacidade do modelo para denoising, estudos e testes mais direcionados devem ser realizados com o propósito de alcançar resultados mais satisfatórios no que concerne a qualidade perceptual e a inteligibilidade da fala em si, considerando os resultados das métricas objetivas. A utilização de um dataset mais robusto com áudios de qualidade avançada, a introdução de ruídos mais variados e naturais, como sons de carro, vento, etc, além do ruído gaussiano, a integração de perdas perceptuais, talvez com a inclusão de modelos pré-treinados especializados em percepção e qualidade de fala, como Wav2Vec ou VGGish, e a utilização de mel-spectrograms como alternativa aos espectrogramas de potência são algumas das possibilidades a serem exploradas em estudos posteriores.

References

[1] Murphy, K. P. (2022). *Probabilistic Machine Learning: An Introduction*. MIT Press.

[2] Murphy, K. P. (2023). *Probabilistic Machine Learning: Advanced Topics*. MIT Press.

[3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] Razavi, A., van den Oord, A., & van der Weele, G. (2019). Vector Quantized Variational Autoencoders. Retrieved from https://arxiv.org/abs/2202.01987

[5] A. van den Oord, O. Vinyals, K. Kavukcuoglu, "Neural Discrete Representation Learning," arXiv preprint arXiv:1711.00937, 2017.

[6] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, "β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.

[7] Goméz-Bombarelli et al. (2017). *Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules*. (DOI: 1610.02415)

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[9] Razavi, A., van den Oord, A., & van der Weele, G. (2019). Vector Quantized Variational Autoencoders. Retrieved from https://arxiv.org/abs/2202.01987

[10] Benesty, J., Schäuble, S., & Springer, M. (2004). Speech Enhancement: Theory and Practice. Retrieved from https://link.springer.com/book/10.1007/3-540-27489-8

[11] Hansen, L. K., & Asari, T. (2009). Noise Reduction in Speech Signals. Retrieved from https://link.springer.com/book/10.1007/978-3-642-00296-0

[12] Tan, J., & Xu, B. (2020). Deep Learning for Speech Enhancement: A Survey. Retrieved from https://www.mdpi.com/2094448

[13] van den Oord, A., & van der Weele, G. (2020). Discrete VAE++: Learning Better Discrete Representations with Mutual Information Maximization. Retrieved from https://arxiv.org/pdf/2205.12248

[14] Chen, J., et al. (2022). VQ-VAE: A Hybrid Model for Speech Enhancement. Retrieved from https://arxiv.org/pdf/2206.15155

[15] Weng, S., et al. (2018). Perceptual Loss for Speech Enhancement. Retrieved from http://apsipa.org/proceedings/2022/APSIPA

[16] Sun, L., et al. (2019). Spectral-Domain Loss for Speech Enhancement. Retrieved from https://arxiv.org/pdf/2206.07293

[17] Li, H., et al. (2023). Multi-Objective Loss for Speech Enhancement in the VQ-VAE Framework. Retrieved from https://arxiv.org/pdf/2010.08115

[18] Liu, X., et al. (2024). Robust Speech Enhancement in Noisy Environments with Multi-Modal VQVAE. Retrieved from https://arxiv.org/pdf/1705.10874

[19] Choi, et al. (2018). Fast Spectrogram Inversion using Multi-Head Convolutional Neural Networks.

[20] Yang, et al. (2023). Guided Speech Enhancement Network.

[21] Hu, J., Shen, L., & Sun, G. (2017). Squeeze-and-Excitation Networks. arXiv preprint arXiv:1709.01507.

[22] Roy, A. G., Siddiqui, S., Pölsterl, S., Navab, N., & Wachinger, C. (2018). Concurrent Spatial and Channel 'Squeeze & Excitation' in Fully Convolutional Networks. arXiv preprint arXiv:1803.02579.