UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE SÃO JOÃO DA BOA VISTA

RAPHAEL PRIZIMICH DE ALMEIDA GAMERO VICO

Sintonização de hiperparâmetros de autocodificadores convolucionais para redução de ruído em camada de mistura

São João da Boa Vista
2025

Raphael Prizimich de Almeida Gamero Vico

Sintonização de hiperparâmetros de autocodificadores convolucionais para redução de ruído em camada de mistura

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Aeronáutica do Campus de São João da Boa Vista, Universidade Estatual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Aeronáutica.

Orientador: Profª Drª. Leandra Isabel de Abreu

São João da Boa Vista
2025

V638s

Vico, Raphael Prizimich de Almeida Gamero
Sintonização de hiperparâmetros de autocodificadores convolucionais para redução de ruído em camada de mistura / Raphael Prizimich de Almeida Gamero Vico. -- São João da Boa Vista, 2025
54 p.

Trabalho de conclusão de curso (Bacharelado - Engenharia Aeronáutica) - Universidade Estadual Paulista (UNESP), Faculdade de Engenharia, São João da Boa Vista
Orientadora: Leandra Isabel de Abreu

1. Inteligência artificial. 2. Dinâmica dos fluidos. 3. Convoluções (Matemática). 4. Ruido. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA - CÂMPUS DE SÃO JOÃO DA BOA VISTA
GRADUAÇÃO EM ENGENHARIA AERONÁUTICA

TRABALHO DE CONCLUSÃO DE CURSO

SINTONIZAÇÃO DE HIPERPARÂMETROS DE AUTOCODIFICADORES CONVOLUCIONAIS PARA REDUÇÃO DE RUÍDO EM CAMADA DE MISTURA

Aluno: Raphael Prizimich de Almeida Gamero Vico
Orientador: Prof.ª Dr.ª Leandra Isabel de Abreu

Banca Examinadora:

- Leandra Isabel de Abreu (Orientadora)
- Gabriel Pereira Gouveia da Silva (Examinador)
- Angelo José Castro Alves Ferreira Filho (Examinador)

Os formulários de avaliação e a ata da defesa, na qual consta a aprovação do trabalho, devidamente assinados pela banca encontram-se no prontuário eletrônico do aluno.

São João da Boa Vista, 28 de novembro de 2025

Dedico este trabalho a todos que nunca desistiram de mim, cuja fé no meu potencial me trouxe até aqui.

AGRADECIMENTOS

Agradeço primeiramente à minha família, base de tudo. Ao meu pai, pela sabedoria e pelos conselhos que guiaram minhas escolhas. À minha mãe, por todo o amor incondicional e pelo suporte constante que me permitiu chegar até aqui. Ao meu irmão, que sempre foi uma fonte de inspiração; ver sua trajetória me motiva a construir a minha.

Expresso minha gratidão à Professora Leandra, não apenas pela oportunidade de desenvolver esta pesquisa, mas pela orientação segura, pela paciência e pela confiança depositada no meu potencial acadêmico.

Aos amigos da República Asa Delta, meu muito obrigado pelo companheirismo, pelas risadas nos momentos de descontração e pelo apoio mútuo durante os desafios da graduação. Vocês tornaram essa jornada mais leve.

Por fim, agradeço a todos que, de alguma forma, fizeram parte da minha trajetória e nunca deixaram de acreditar em mim.

"Não entre em pânico."
(Douglas Adams)

RESUMO

Este trabalho apresenta a sintonização de hiperparâmetros de *autoencoders* convolucionais para a redução de ruído em imagens provenientes de simulações numéricas de escoamento em camada de mistura. Utilizando uma base de dados contendo imagens com diferentes níveis de ruído artificialmente adicionados, aplicou-se o *framework* Optuna para otimizar parâmetros da rede, como número de filtros, tamanho do *kernel*, taxa de aprendizado e funções de ativação. Os resultados demonstraram eficácia na remoção de ruídos para a maioria dos níveis testados, embora para ruídos de alta intensidade (0 dB) a remoção tenha sido parcial. A análise da matriz de reconstrução indicou uma capacidade de generalização quando modelos treinados em condições mais ruidosas foram aplicados a imagens menos ruidosas. O *autoencoder* mostrou-se uma ferramenta promissora para melhoria da qualidade dos dados experimentais em fluidos. Para trabalhos futuros, recomenda-se a aplicação da métrica *Structural Similarity Index* (SSIM) para avaliação qualitativa das imagens reconstruídas, além da exploração de outras bases de dados e técnicas de otimização.

PALAVRAS-CHAVE: autoencoder convolucional; remoção de ruído; sintonização de hiperparâmetros; simulação numérica; camada de mistura.

ABSTRACT

This work presents the hyperparameter tuning of convolutional autoencoders for noise reduction in images derived from numerical simulations of mixing layer flow. Using a dataset containing images with different levels of artificially added noise, the Optuna framework was applied to optimize network parameters, such as the number of filters, kernel size, learning rate, and activation functions. The results demonstrated effectiveness in noise removal for most tested levels, although for high-intensity noise (0 dB), the removal was partial. The analysis of the reconstruction matrix indicated a generalization capability when models trained in noisier conditions were applied to less noisy images. The autoencoder proved to be a promising tool for improving the quality of experimental fluid data. For future work, the application of the Structural Similarity Index (SSIM) metric is recommended for the qualitative assessment of reconstructed images, in addition to the exploration of other datasets and optimization techniques.

KEYWORDS: convolutional autoencoder; noise removal; hyperparameter tuning; numerical simulation; mixing layer.

LISTA DE ILUSTRAÇÕES

Figura 1 – Esquema conceitual de um *Perceptron*. 14
Figura 2 – Função sigmoid. 15
Figura 3 – Função tanh. 16
Figura 4 – Função relu. 17
Figura 5 – Função linear. 17
Figura 6 – Função Leaky ReLU. 18
Figura 7 – Função Softplus. 19
Figura 8 – Função Softsign. 19
Figura 9 – Função SELU. 20
Figura 10 – Função ELU. 21
Figura 11 – Função Exponencial. 21
Figura 12 – Função Swish. 22
Figura 13 – Função GELU. 23
Figura 14 – Decorrências de diferentes magnitudes para a taxa de aprendizado. 25
Figura 15 – Gráfico comparando a convergência da função de custo (L) ao longo do tempo (t) 26
Figura 16 – Esquema conceitual de *perceptron* de múltiplas camadas. 27
Figura 17 – Esquema conceitual de Redes Neurais Recorrentes. 27
Figura 18 – Esquema conceitual de redes neurais convolucionais. 28
Figura 19 – Ilustração visual dos conceitos de *underfitting*, *overfitting* e *balanced* (Generalização). 30
Figura 20 – Exemplo de uma imagem da simulação. 33
Figura 21 – (a) Imagem Original, (b) Imagem com um único canal de cor. 33
Figura 22 – (a) Histograma da imagem original, (b) Histograma da imagem normalizada. . . . 34
Figura 23 – (a) Imagem original, (b) Imagem normalizada. 34
Figura 24 – Exemplos de transformações de *data augmentation*: (a) Rotação, (b) Zoom, (c) Deslocamento de Largura, (d) Deslocamento de Altura, (e) Inversão Horizontal, e (f) Inversão Vertical. 35
Figura 25 – a) Imagem com SNR $30dB$; b) Imagem com SNR$15dB$; c) Imagem com SNR $5dB$; d) Imagem com SNR $0dB$ 37
Figura 26 – Exemplo de um *autoencoders* convolucionais. 37
Figura 27 – Evolução dos parâmetros estruturais (filtros, kernel e camadas) em relação à SNR. 41
Figura 28 – Dinâmica de otimização (*learning rate* e *batch size*) em função da SNR. 43
Figura 29 – Seleção das funções de ativação, tamanho de *pooling* e função de custo em relação à SNR. 44
Figura 30 – Evolução da função de custo (*loss*) para os diferentes níveis de SNR. 46
Figura 31 – Comparação das curvas de MSE durante o treinamento. 47
Figura 32 – Comparação das curvas de MAE durante o treinamento. 48
Figura 33 – Curvas de custo de treinamento e validação para cada cenário de SNR. 49

Figura 34 – Matriz de reconstrução do escoamento. As linhas indicam o modelo utilizado e as colunas o nível de ruído da imagem de entrada. 50
Figura 35 – Mapas de erro absoluto. A intensidade luminosa representa a magnitude da discrepância local entre a imagem original e a predita. 51
Figura 36 – Matriz de desempenho (RMSE) dos modelos treinados sob diferentes condições de ruído. 52

SUMÁRIO

1 INTRODUÇÃO 12

2 OBJETIVOS 13

3 FUNDAMENTAÇÃO TEÓRICA 14

3.1 Fundamentos das redes neurais artificiais 14

3.1.1 Funções de Ativação 15

3.1.2 Funções de Custo 23

3.1.3 Funções de otimização 24

3.1.4 Taxa de aprendizado 25

3.1.5 Tamanho de lote 25

3.2 Arquiteturas de Redes Neurais 26

3.2.1 Perceptron de Múltiplas Camadas (MLP) 26

3.2.2 Redes Neurais Recorrentes (RNN) 27

3.2.3 Redes Neurais Convolucionais (CNN) 27

3.3 Treinamento e Validação 29

3.3.1 Preparação e Normalização dos Dados 29

3.3.2 Divisão em Treinamento, Validação e Teste 29

3.3.3 *Overfitting*, *Underfitting* e Generalização 30

3.4 Fundamentos da Camada de Mistura 31

3.4.1 Microescala de Taylor e o Número de Reynolds 31

3.4.2 Simulação Numérica Direta (DNS) 31

4 METODOLOGIA 33

4.1 Base de dados 33

4.1.1 Pré-processamento de dados 33

4.1.2 Aumento de dados 35

4.1.3 Adição de Ruído 35

4.2 Arquitetura do *Autoencoder* Convolucional 37

4.3 Divisão de Amostras de Treino e Teste 38

4.4 Sintonização de Hiperparâmetros 38

5 RESULTADOS E DISCUSSÕES 40

5.1 Resultados da Sintonização de Hiperparâmetros 40

5.1.1 Painel Estrutural - Topologia e Capacidade de Compressão 40

5.1.2 Painel de Otimização - Estabilidade e Convergência 42

5.1.3 Painel Qualitativo - Função de ativação (interna e saída), Função de custo (*loss*) e Fator de *upsampling*. 44

5.2 Desempenho do Treinamento e Análise de Convergência 45
5.2.1 Convergência e Estabilidade por Nível de Ruído 45
5.2.2 Generalização e o Efeito do Gargalo 48
5.3 Avaliação de Robustez e Generalização Cruzada 49
5.3.1 Reconstrução Visual e Topologia do Escoamento 50
5.3.2 Análise Quantitativa do Erro de Reconstrução 51

6 CONCLUSÃO 53

REFERÊNCIAS 54

1 INTRODUÇÃO

A mecânica dos fluidos é o ramo da física que estuda o comportamento dos fluidos, tanto em repouso (estática dos fluidos) quanto em movimento (dinâmica dos fluidos). Os fluidos, que incluem líquidos e gases, possuem uma vasta gama de aplicações na engenharia, sendo essenciais em áreas como aeronáutica, hidráulica e processos industriais (White, 1988).

Para analisar sistemas complexos que envolvem escoamento, transferência de calor e fenômenos correlatos, a Dinâmica dos Fluidos Computacional (termo em inglês *Computational Fluid Dynamics*, CFD) surge como uma ferramenta numérica complementar à teoria clássica. Baseada nas equações de Naver-Stokes, o CFD realiza simulações numéricas para investigar esses fenômenos de maneira eficiente, porém ainda com custo computacional elevado (White, 1988). Outra maneira de estudar escoamentos é através de experimentos, que leva a resultados ainda mais precisos. Embora experimentos sejam valiosos, eles estão sujeitos a limitações práticas, como ruídos em instrumentos de medição, como por exemplo, imprecisões em sensores ou câmeras, que podem prejudicar a qualidade dos resultados. A remoção desses ruídos é, portanto, um desafio crucial (Ashgriz; Mostaghimi, 2002).

Neste contexto, uma das técnicas que apresenta alto potencial para a tarefa de filtragem de dados são os autocodificadores, ou em inglês *autoencoders*, mais especificamente quando se trata de base de dados de imagens, é mais eficiente o uso de *autoencoders* convolucionais. Tal técnica trata-se de um tipo de rede neural que emprega camadas convolucionais para comprimir imagens em uma representação latente (*encoder*) e, em seguida, reconstruí-las (*decoder*). Essa arquitetura é especialmente eficaz na captura de padrões espaciais e hierárquicos, permitindo não apenas a eliminação de ruídos, mas também a preservação de detalhes essenciais durante a reconstrução (Goodfellow *et al.*, 2016).

O presente estudo utiliza a técnica de *autoencoder* convolucional para remover ruído de diferentes intensidades em imagens de escoamento de fluido. A base de dados utilizada foi os resultados de uma simulação numérica DNS de uma camada de mistura com evolução espacial em Re_λ 250, do estudo de Fiscaletti *et al.* (2016). Foi inserido ruído artificialmente nas imagens da base de dados. O treinamento da rede visa a reconstrução das imagens para suas versões limpas, a partir da diminuição da dimensionalidade no espaço latente. Para isso, investiga-se como diferentes intensidades de ruído demandam ajustes específicos em hiperparâmetros da rede neural, como taxa de aprendizagem, número de camadas e funções de ativação. O objetivo principal é garantir o melhor desempenho possível na remoção de componentes espúrios, preservando as estruturas características do escoamento.

A estrutura deste trabalho é composta por seis capítulos. O Capítulo 2 define os objetivos da pesquisa, detalhando o objetivo principal e os específicos. Em seguida, o Capítulo 3 estabelece a base teórica sobre redes neurais e camada de mistura. O Capítulo 4 descreve toda a metodologia, desde a preparação dos dados até a implementação do *autoencoder* e a otimização de seus hiperparâmetros. O Capítulo 5 apresenta e analisa os resultados obtidos tanto na sintonização quanto na remoção de ruído. Finalmente, o Capítulo 6 oferece as conclusões e as perspectivas para futuras investigações.

2 OBJETIVOS

O objetivo principal deste estudo é realizar a sintonização de hiperparâmetros de autocodificadores convolucionais para reduzir o ruído presente em simulações numéricas de escoamento de uma camada de mistura. O foco está em avaliar como a escolha dos hiperparâmetros do modelo influencia no desempenho da remoção de ruído, visando preservar as estruturas físicas relevantes do escoamento e melhorar a qualidade dos dados simulados para análises posteriores.

Para alcançar este objetivo principal, os seguintes objetivos específicos foram definidos:

- Realizar uma revisão bibliográfica sobre aprendizado de máquina, com foco em redes neurais convolucionais e autocodificadores.
- Estudar e analisar a base de dados do escoamento da camada de mistura (Fiscaletti *et al.*, 2016), calculando estatísticas, extraindo histogramas e gerando espectrogramas.
- Implementar o modelo de autocodificador convolucional em Python para extrair os primeiros resultados de remoção de ruído.
- Ajustar os hiperparâmetros da rede neural para otimizar o desempenho do modelo, utilizando a biblioteca Optuna (Akiba *et al.*, 2019) para automatizar a otimização.
- Compilar e analisar os resultados obtidos após a otimização.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, é apresentada a fundamentação teórica que embasa este trabalho. A discussão é iniciada pelos fundamentos das redes neurais artificiais (Seção 3.1), detalhando seus componentes cruciais — funções de ativação, de custo e de otimização. Em seguida, são exploradas as diferentes arquiteturas de redes neurais (Seção 3.2), bem como os métodos para o treinamento e validação dos modelos (Seção 3.3). Por fim, apresenta-se a base teórica referente à camada de mistura (Seção 3.4), contextualizando o fenômeno físico estudado.

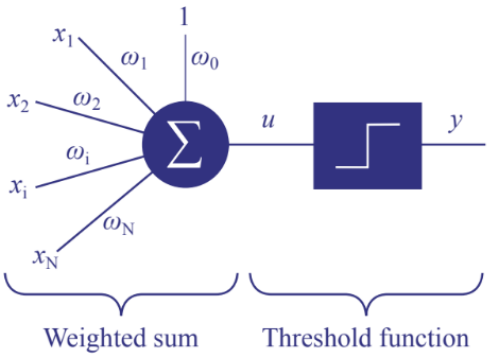
3.1 Fundamentos das redes neurais artificiais

Redes neurais artificiais (RNAs) são modelos computacionais inspirados na estrutura e funcionamento do cérebro humano. Elas são compostas por unidades de processamento interconectadas, chamadas neurônios, organizadas em camadas. A principal capacidade das RNAs é aprender padrões e relações diretamente a partir de dados para realizar tarefas complexas.

Essas tarefas são frequentemente divididas em duas categorias principais: *classificação*, onde o modelo aprende a atribuir um rótulo categórico a uma entrada (por exemplo, identificar se um *e-mail* é *spam* ou não), e *regressão*, onde o modelo prevê um valor contínuo (como estimar o preço de um imóvel ou a temperatura) (Goodfellow *et al.*, 2016). O problema estudado no presente trabalho se trata de um problema de regressão.

A unidade fundamental de uma rede neural é o *perceptron*, a forma mais simples de RNA (1). O perceptron consiste em um único neurônio que recebe um ou mais dados de entrada. Cada entrada é multiplicada por um peso (w), que quantifica a importância dessa conexão. Em seguida, os valores ponderados são somados, e um parâmetro adicional, o *bias* (b), é adicionado a essa soma. O *bias* permite que a função de ativação seja deslocada (transladada), aumentando a flexibilidade do modelo para se ajustar aos dados. Por fim, esse resultado passa por uma função de ativação para gerar a saída (Haykin, 1994).

Figura 1 – Esquema conceitual de um *Perceptron*.



Fonte: Próprio autor

Embora eficaz para problemas linearmente separáveis, o *Perceptron* simples não consegue modelar relações complexas. Para superar essa limitação, utiliza-se o *Perceptron* Multicamadas (MLP). Um MLP é uma arquitetura mais profunda que possui pelo menos três tipos de camadas: uma camada de entrada (que recebe os dados brutos), uma ou mais camadas ocultas e uma camada de saída (que produz o resultado final). São as camadas ocultas que permitem ao MLP aprender características e padrões não-lineares complexos (Haykin, 1994). O funcionamento de um MLP é governado por seus componentes principais, como as funções de ativação, que introduzem não-linearidade, e as funções de custo, que medem o erro do modelo.

3.1.1 Funções de Ativação

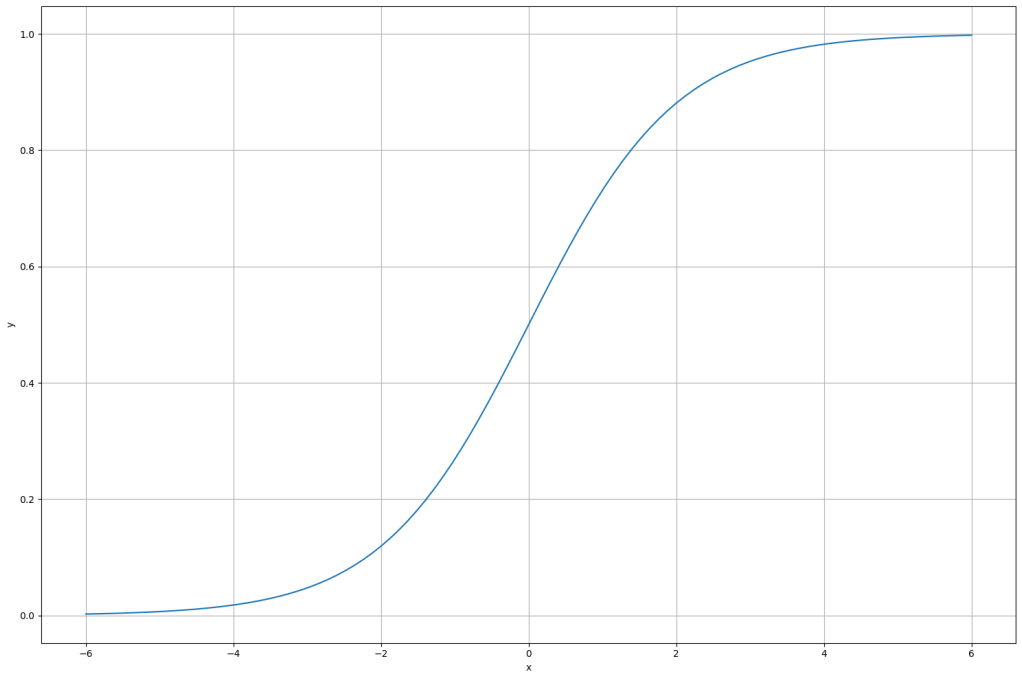
As funções de ativação são um componente essencial das redes neurais, responsáveis por introduzir não-linearidade no modelo. Sem elas, mesmo um MLP profundo se comportaria apenas como um modelo linear simples. Elas são aplicadas à saída ponderada de cada neurônio. A escolha da função de ativação impacta diretamente o desempenho e a estabilidade do treinamento (Haykin, 1994).

Abaixo são descritas algumas das funções de ativação mais utilizadas:

- **Sigmoid**: Também conhecida como função logística. Produz uma saída no intervalo $[0, 1]$ (2).

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

Figura 2 – Função sigmoid.

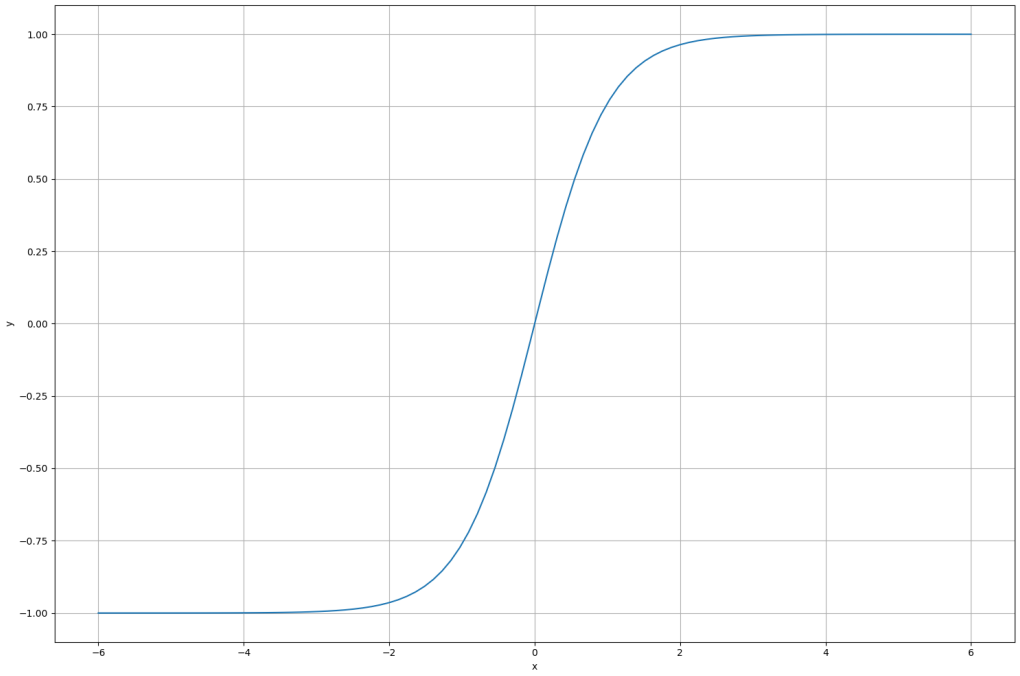


Fonte: Próprio autor

- **Tangente Hiperbólica (Tanh)**: Produz uma saída no intervalo $[-1, 1]$ (3).

$$f(x) = \tanh(x) \tag{2}$$

Figura 3 – Função tanh.

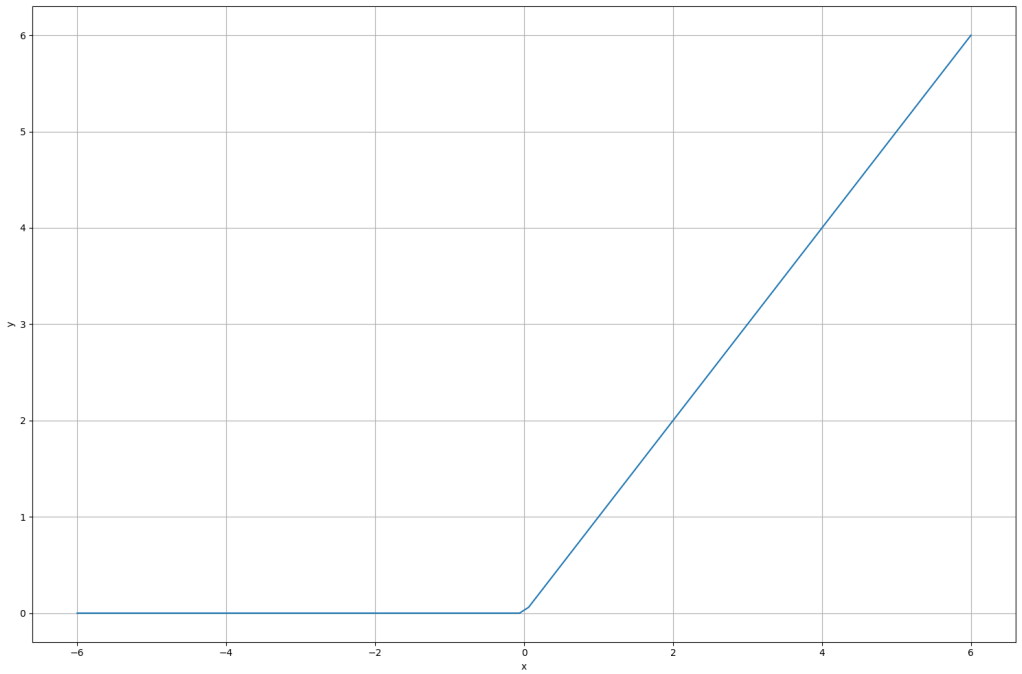


Fonte: Próprio autor

- **ReLU (Unidade Linear Retificada)**: Função mais popular para camadas ocultas. Produz uma saída no intervalo $[0, \infty)$ (4).

$$f(x) = \max(0, x) \tag{3}$$

Figura 4 – Função relu.

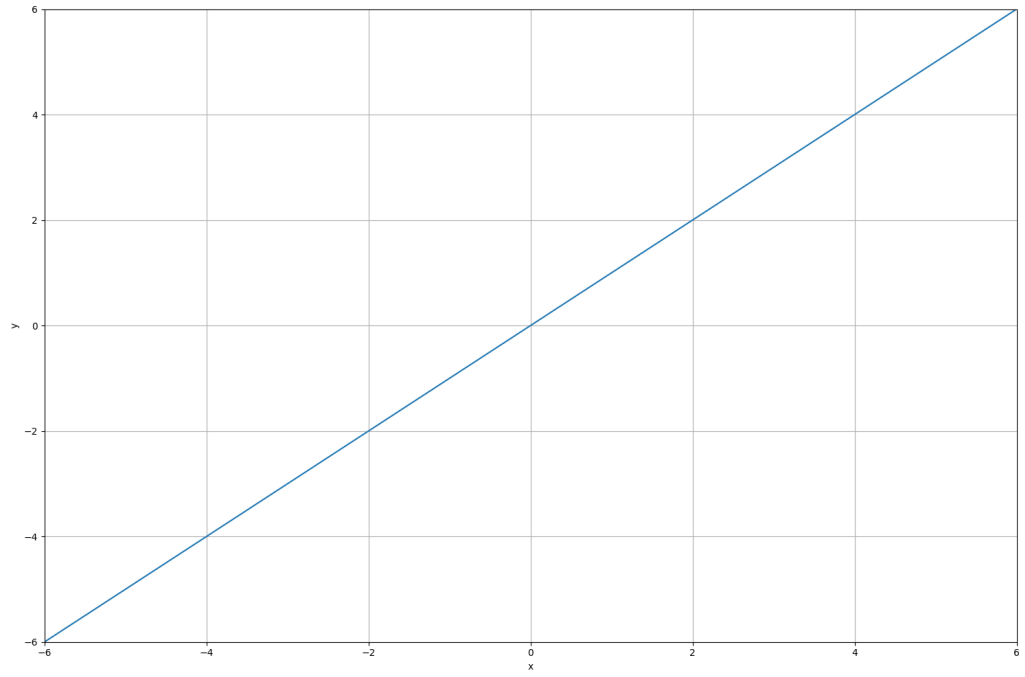


Fonte: Próprio autor

- **Linear (Identidade)**: Função utilizada na camada de saída de redes para regressão. Produz uma saída no intervalo $(-\infty, \infty)$ (4).

$$f(x) = x \tag{4}$$

Figura 5 – Função linear.



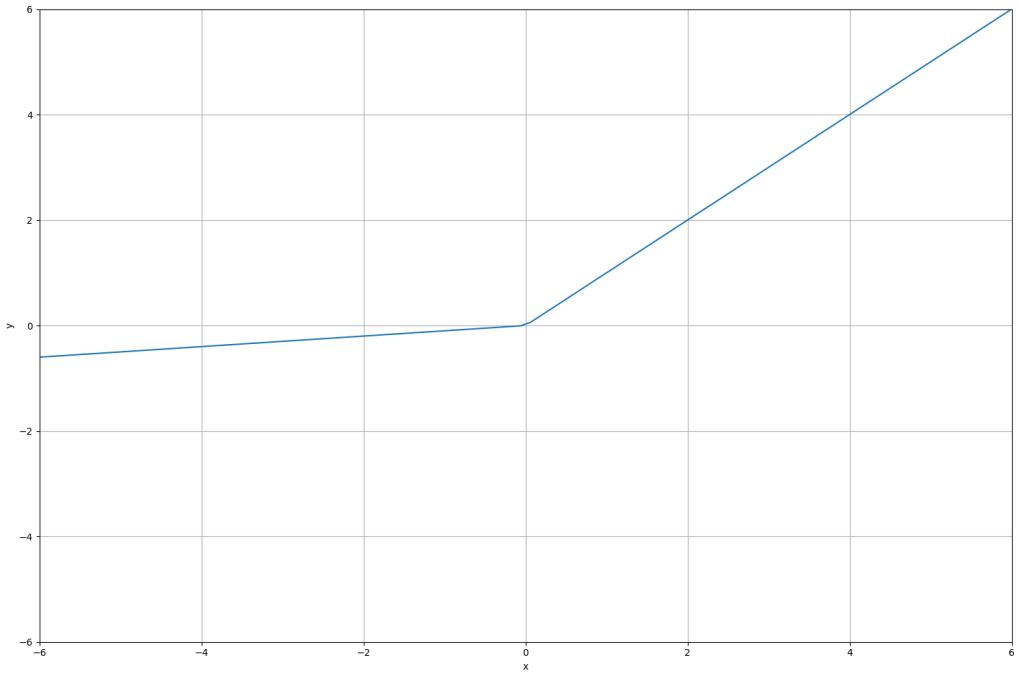
Fonte: Próprio autor

- **Leaky ReLU**: Uma variação da ReLU que permite um pequeno gradiente não nulo (α) para

valores negativos, evitando o problema de "neurônios mortos". Produz uma saída no intervalo $(-\infty, \infty)$ (6).

$$f(x) = \begin{cases} x & \text{se } x > 0 \\ \alpha x & \text{se } x \leq 0 \end{cases} \tag{5}$$

Figura 6 – Função Leaky ReLU.

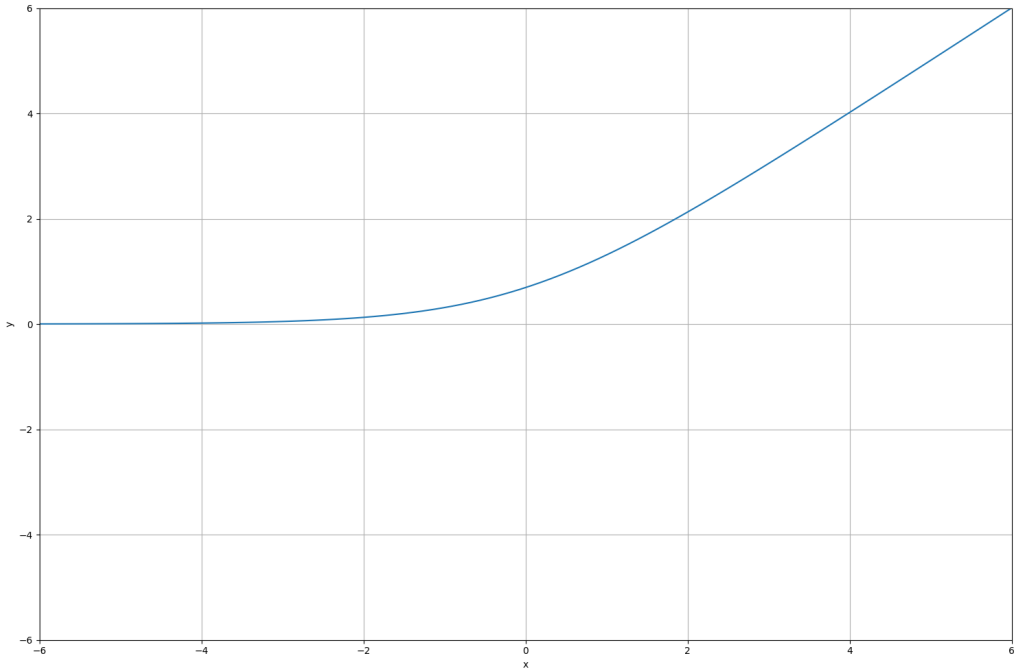


Fonte: Próprio autor

- **Softplus**: Uma aproximação suave e diferenciável da função ReLU. Produz uma saída no intervalo $(0, \infty)$ (7).

$$f(x) = \ln(1 + e^x) \tag{6}$$

Figura 7 – Função Softplus.

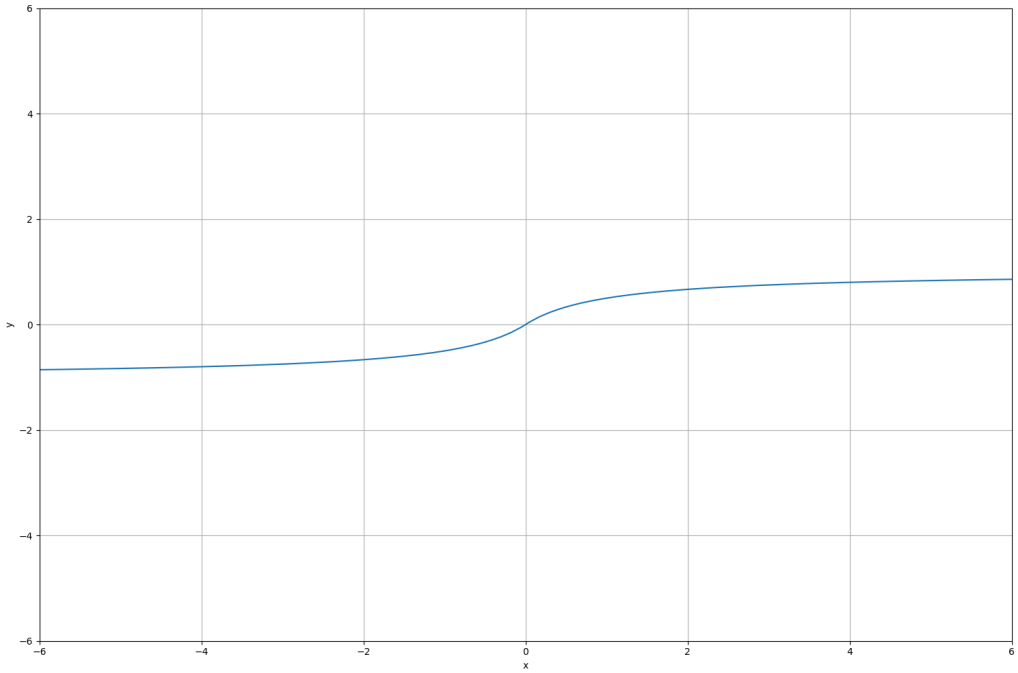


Fonte: Próprio autor

- **Softsign**: Semelhante à Tanh, mas converge polinomialmente em vez de exponencialmente para as assíntotas. Produz uma saída no intervalo $(-1, 1)$ (8).

$$f(x) = \frac{x}{1 + |x|} \tag{7}$$

Figura 8 – Função Softsign.

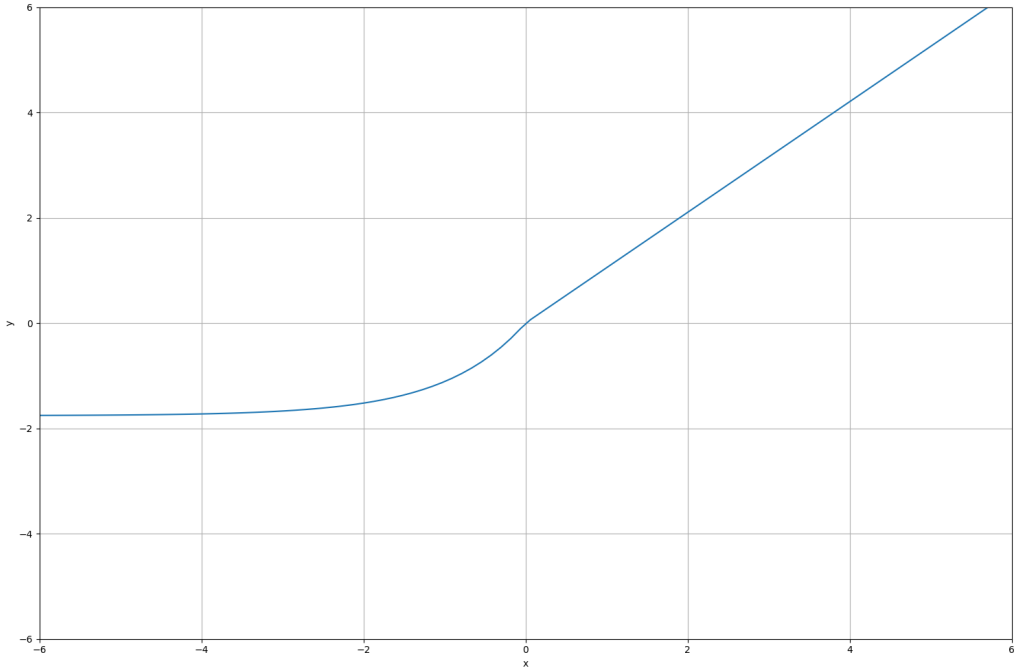


Fonte: Próprio autor

- **SELU (Unidade Linear Exponencial Escalada)**: Induz propriedades de auto-normalização na rede neural. Possui parâmetros de escala (λ) e alfa (α) fixos (9).

$$f(x) = \lambda \begin{cases} x & \text{se } x > 0 \\ \alpha(e^x - 1) & \text{se } x \leq 0 \end{cases} \tag{8}$$

Figura 9 – Função SELU.

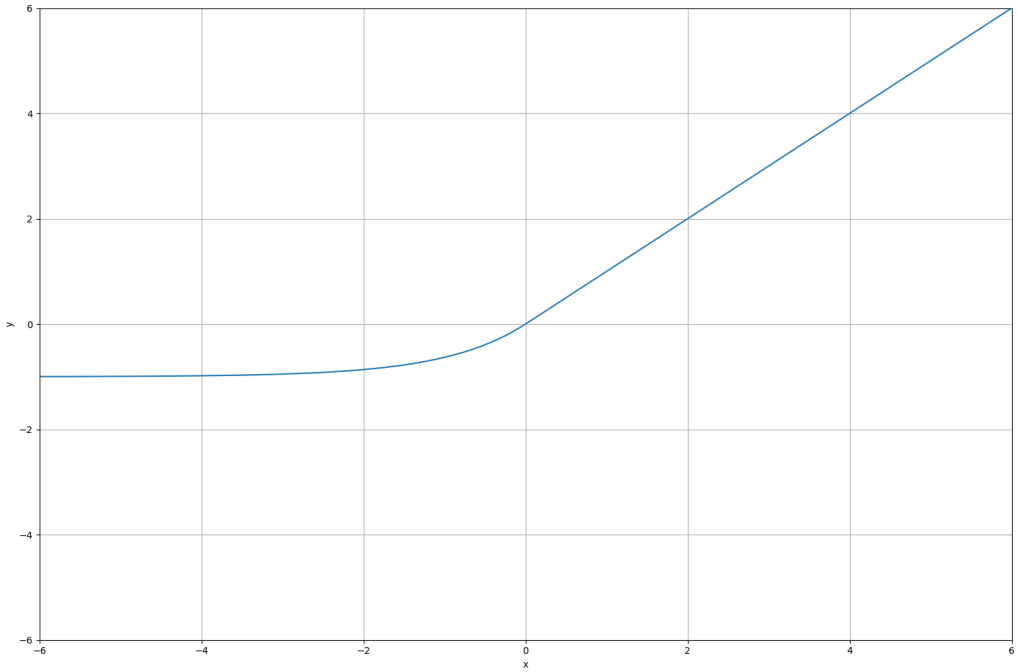


Fonte: Próprio autor

- **ELU (Unidade Linear Exponencial)**: Similar à ReLU, mas com valores negativos suaves que ajudam a empurrar a média das ativações para zero, acelerando o aprendizado (10).

$$f(x) = \begin{cases} x & \text{se } x > 0 \\ \alpha(e^x - 1) & \text{se } x \leq 0 \end{cases} \tag{9}$$

Figura 10 – Função ELU.

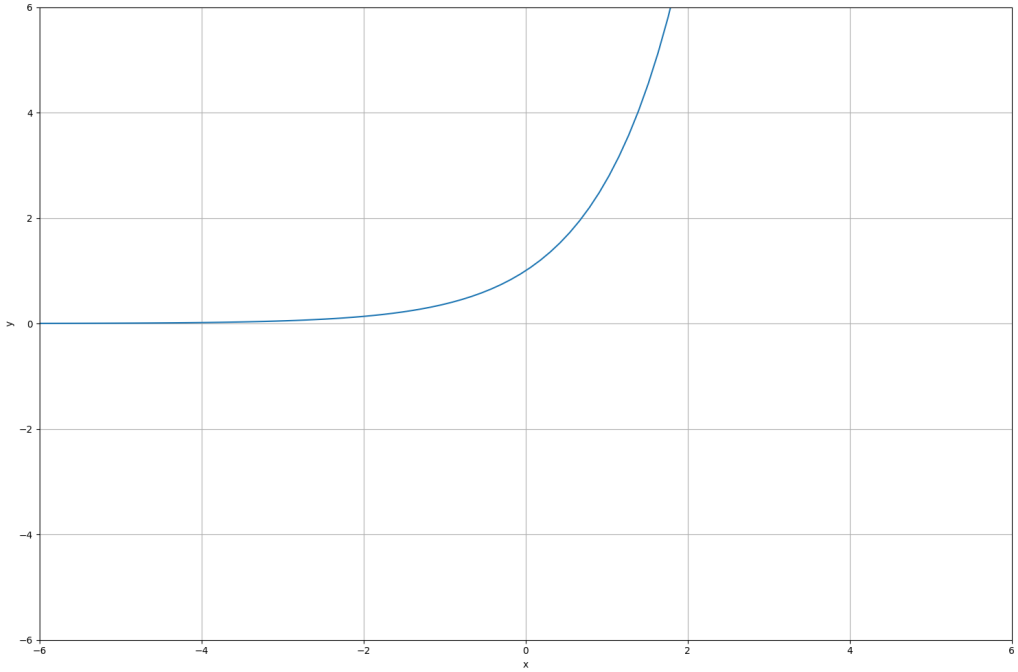


Fonte: Próprio autor

- **Exponencial**: Aplica a função exponencial base e. Pode resultar em valores muito altos rapidamente (explosão de gradiente) se não controlada (11).

$$f(x) = e^x \tag{10}$$

Figura 11 – Função Exponencial.

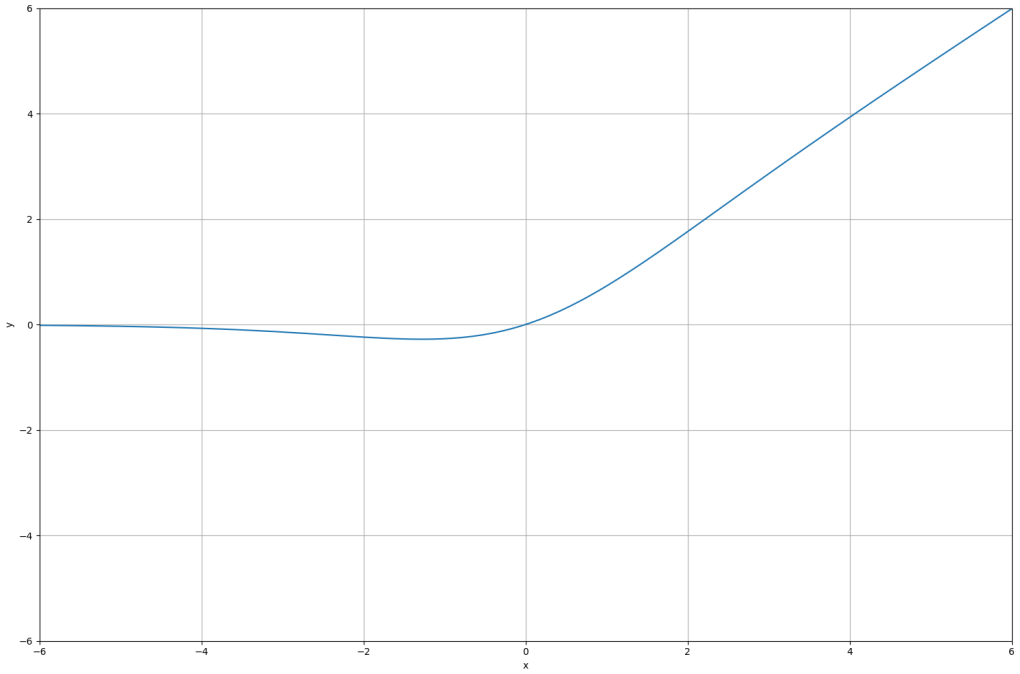


Fonte: Próprio autor

- **Swish**: Uma função não-monotônica proposta pelo Google que tende a funcionar melhor que a ReLU em redes profundas. É definida como x multiplicado pela sigmoide (12).

$$f(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}} \tag{11}$$

Figura 12 – Função Swish.

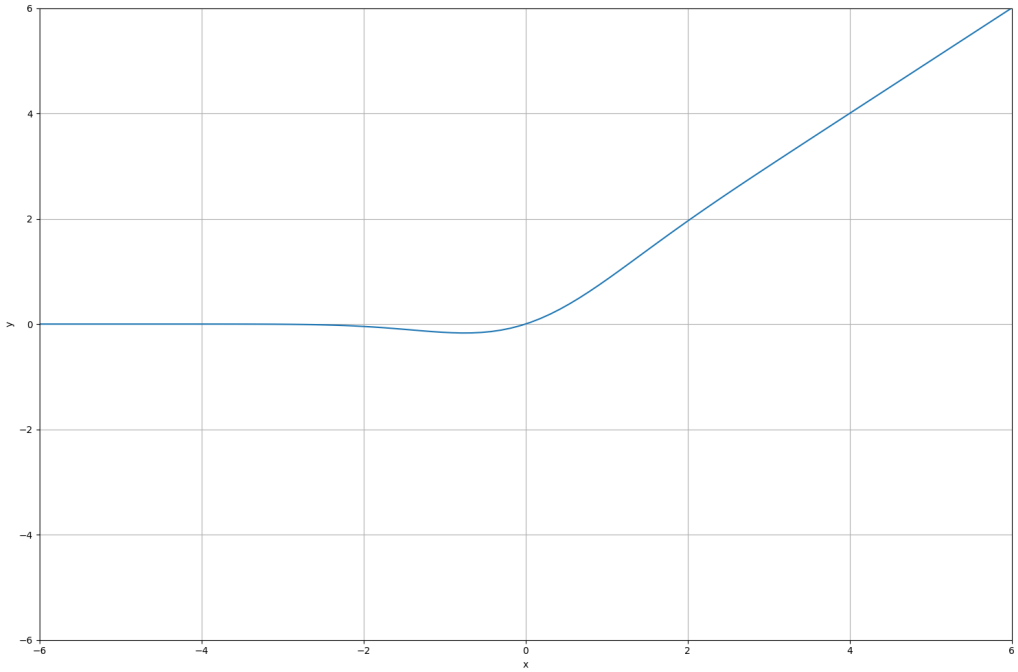


Fonte: Próprio autor

- **GELU (Unidade Linear de Erro Gaussiano)**: Utilizada em arquiteturas modernas como BERT e GPT. Pondera as entradas pela sua magnitude usando a função de erro (13).

$$f(x) = 0.5x \left(1 + \tanh\left[\sqrt{2/\pi}(x + 0.044715x^3)\right]\right) \tag{12}$$

Figura 13 – Função GELU.



Fonte: Próprio autor

3.1.2 Funções de Custo

A função de custo (*loss function*) é a métrica utilizada para avaliar o desempenho do modelo durante o treinamento. Ela quantifica a discrepância entre os valores previstos pela rede e os valores reais. O objetivo de todo o processo de treinamento é ajustar os pesos e *biases* da rede de forma a minimizar o valor dessa função (Haykin, 1994).

A escolha da função de custo está diretamente ligada ao tipo de problema. Para problemas de regressão, as mais comuns são:

- **Erro Quadrático Médio (MSE)**: É a função de custo padrão para regressão. Ela calcula a média das diferenças ao quadrado entre os valores reais (y_i) e os valores previstos ($\hat{y}_i$). Esta função penaliza erros grandes de forma mais significativa.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{13}$$

- **Erro Absoluto Médio (MAE)**: Calcula a média das diferenças absolutas entre os valores reais (y_i) e os previstos ($\hat{y}_i$). Ao não elevar o erro ao quadrado, o MAE é menos sensível a valores atípicos no conjunto de dados.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{14}$$

- **Erro Percentual Absoluto Médio (MAPE)**: Expressa o erro como uma porcentagem média em relação aos valores reais. É uma métrica de fácil interpretação, pois é independente da escala

dos dados, porém torna-se indefinida se $y_i = 0$.

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{15}$$

- **Perda de Poisson**: É uma função de custo baseada na verossimilhança logarítmica, frequentemente utilizada quando os dados de saída seguem uma distribuição de Poisson (como contagens de eventos). Ela penaliza a divergência entre a distribuição prevista e a real.

$$Loss_{Poisson} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i \ln(\hat{y}_i)) \tag{16}$$

Onde n é o número total de amostras, y_i é o valor real da i-ésima amostra e $\hat{y}_i$ é o valor previsto pelo modelo.

3.1.3 Funções de otimização

A função de otimização refere-se ao algoritmo utilizado para ajustar os pesos e *biases* do modelo de forma a minimizar a função de custo. Embora o método base seja o gradiente descendente estocástico (SGD), diversos otimizadores mais avançados foram desenvolvidos para acelerar a convergência e melhorar a estabilidade do treinamento.

Esses otimizadores adaptam a taxa de aprendizado ou utilizam informações de gradientes passados.

- ***Momentum***: Este método busca acelerar o gradiente descendente ao incorporar o conceito de *momentum* (inércia). Ele acumula uma média móvel exponencial dos gradientes passados e utiliza essa média para direcionar a atualização dos parâmetros. Isso resulta em uma aceleração da convergência ao longo de direções onde o gradiente aponta consistentemente, ao mesmo tempo em que atenua oscilações em outras direções. O método é eficaz para navegar mais rapidamente por platôs (regiões de gradiente baixo) e superar mínimos locais rasos (Rumelhart; Hinton; Williams, 1986).

- **RMSProp (*Root Mean Square Propagation*)**: É um método de taxa de aprendizado adaptativo. O RMSProp ajusta a taxa de aprendizado de cada parâmetro individualmente, dividindo-a por uma média móvel da magnitude recente dos gradientes (especificamente, a raiz quadrada da média dos gradientes ao quadrado). Na prática, isso diminui a taxa de aprendizado para parâmetros que recebem gradientes grandes e a aumenta para parâmetros com gradientes pequenos, melhorando a eficiência da convergência (Kingma; Ba, 2014).

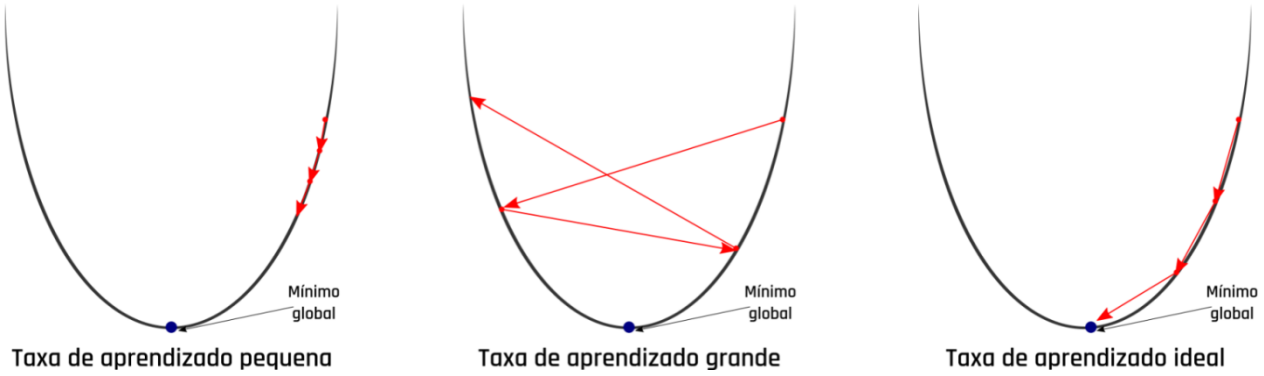
- **ADAM (*Adaptive Moment Estimation*)**: Conforme o nome sugere, este otimizador combina as vantagens dos métodos de *momentum* e *RMSProp*. Ele armazena duas médias móveis: o "primeiro momento"(a média dos gradientes, como o *momentum*) e o "segundo momento"(a média dos gradientes ao quadrado, como o *RMSProp*). O método é “computacionalmente eficiente, possui poucos requisitos de memória, é invariante à reescalação diagonal dos gradientes

e é bem adequado para problemas que são grandes em termos de dados/parâmetros" (Kingma; Ba, 2014).

3.1.4 Taxa de aprendizado

A taxa de aprendizado (*learning rate*) controla o tamanho dos passos que o otimizador dá ao ajustar os pesos do modelo durante o treinamento. Valores maiores podem acelerar a convergência, mas correm o risco de ultrapassar o mínimo da função de custo, enquanto valores muito pequenos podem tornar o treinamento lento e propenso a ficar preso em mínimos locais (14) (Goodfellow *et al.*, 2016).

Figura 14 – Decorrências de diferentes magnitudes para a taxa de aprendizado.



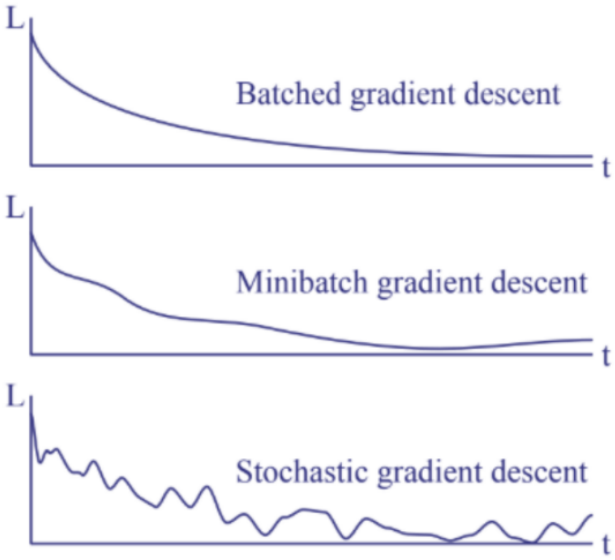
Fonte: Próprio autor

3.1.5 Tamanho de lote

O tamanho do lote (*batch size*) define quantas amostras são processadas antes de atualizar os pesos do modelo, sendo um hiperparâmetro de otimização. Conforme ilustrado na 15, existem três abordagens principais:

- **Gradiente descendente em lote (*batched gradient descent*):** Utiliza o conjunto de treinamento completo para calcular o gradiente. Isso resulta em uma convergência suave (baixa variação no custo L), mas computacionalmente mais lenta por atualização.
- **Gradiente em minilote (*minibatch gradient*):** O conjunto de treinamento é dividido em lotes menores (minilotes), e o gradiente é calculado a cada lote. Oferece um equilíbrio entre a suavidade do método em lote e a rapidez do estocástico.
- **Gradiente estocástico (*stochastic gradient*):** Calcula o gradiente para cada amostra individual (tamanho de lote 1). Isso leva a atualizações muito frequentes e uma convergência ruidosa, como visto na curva de custo (L) oscilante.

Figura 15 – Gráfico comparando a convergência da função de custo (L) ao longo do tempo (t)



Fonte: Próprio autor

Lotes menores (*mini batch* ou Estocástico) proporcionam atualizações mais frequentes, o que pode levar a um treinamento mais rápido, mas também uma convergência mais ruidosa. Lotes maiores (como o *batched*) oferecem estimativas de gradiente mais precisas, mas exigem mais memória e podem levar a uma convergência mais lenta (Goodfellow *et al.*, 2016).

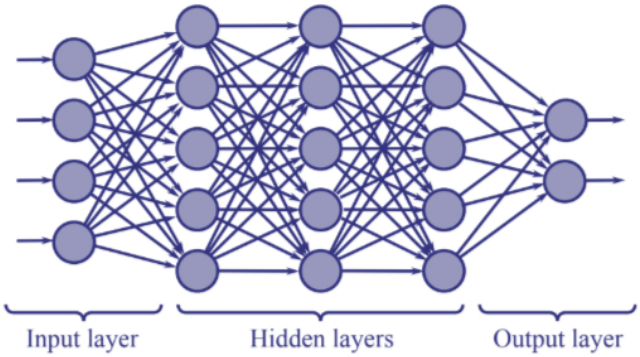
3.2 Arquiteturas de Redes Neurais

As redes neurais podem ser estruturadas em diferentes arquiteturas, cada uma especializada em um tipo de dado ou tarefa. Embora existam muitas variações, as mais fundamentais incluem o perceptron de múltiplas camadas (MLP), as redes neurais recorrentes (RNN) e as redes neurais convolucionais (CNN).

3.2.1 Perceptron de Múltiplas Camadas (MLP)

O *perceptron* de múltiplas camadas (MLP), já introduzido na seção 3.1, é a arquitetura de rede neural mais clássica. Ela consiste em uma sequência de camadas onde cada neurônio de uma camada está conectado a todos os neurônios da camada seguinte (16). Devido a essa estrutura de conexão completa, essas camadas são conhecidas como camadas totalmente conectadas (*fully-connected*) ou, comumente, camadas densas (*dense layers*). Esta arquitetura é amplamente utilizada para problemas de regressão e classificação com dados tabulares ou estruturados, onde as relações espaciais ou temporais entre as entradas não são predominantes (Goodfellow *et al.*, 2016).

Figura 16 – Esquema conceitual de *perceptron* de múltiplas camadas.



Fonte: Próprio autor

3.2.2 Redes Neurais Recorrentes (RNN)

As redes neurais recorrentes (RNNs) são projetadas para lidar com dados sequenciais, como séries temporais, texto (processamento de linguagem natural) ou áudio. Sua característica principal é a presença de conexões recorrentes (17), isto é, conexões que formam ciclos. Essas conexões permitem que a rede mantenha um estado interno ou memória das informações processadas anteriormente, tornando-a capaz de identificar padrões e dependências ao longo do tempo, algo que arquiteturas não-recorrentes, como o MLP, não conseguem (Goodfellow *et al.*, 2016).

Figura 17 – Esquema conceitual de Redes Neurais Recorrentes.



Fonte: Próprio autor

3.2.3 Redes Neurais Convolucionais (CNN)

As Redes Neurais Convolucionais (CNNs) são a arquitetura dominante para tarefas de visão computacional, como redução de ruído, segmentação de imagens e detecção de objetos. Inspiradas no funcionamento do córtex visual humano, as CNNs diferem fundamentalmente dos MLPs por sua capacidade de processar os dados mantendo sua estrutura espacial original (ex: uma grade de *pixels*). Em contraste, os MLPs requerem que a entrada seja vetorizada o que descarta as informações sobre as relações espaciais entre os *pixels* (Goodfellow *et al.*, 2016).

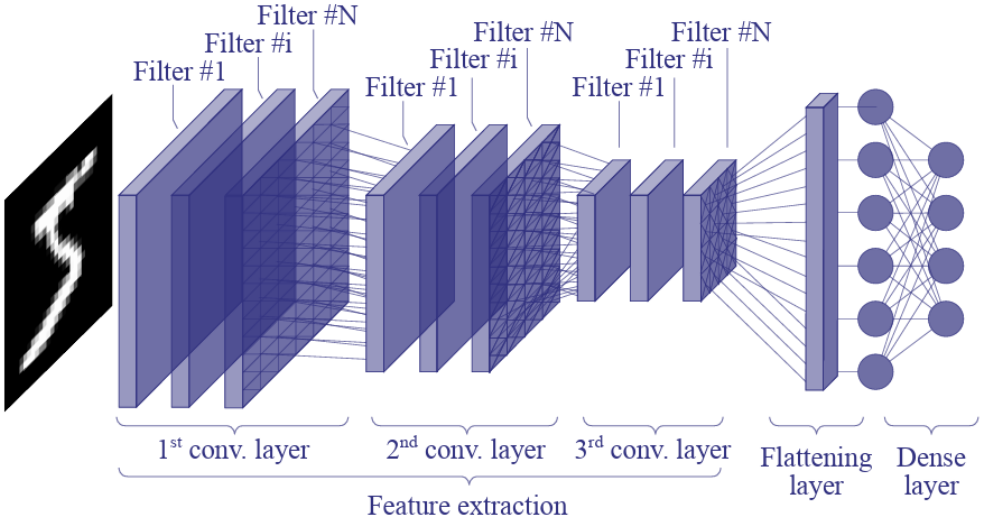
A principal inovação da CNN é a camada convolucional. Esta camada opera aplicando um conjunto de filtros, também conhecidos como *kernels*, para realizar operações de convolução sobre a entrada

(18). Um *kernel* é um pequeno conjunto de pesos que desliza sobre a imagem para identificar padrões locais. Este processo permite duas vantagens cruciais:

1. **Extração de características locais:** O filtro identifica padrões locais, como bordas, texturas ou formas.

2. **Compartilhamento de parâmetros:** O mesmo filtro é reutilizado em toda a extensão da imagem, detectando o mesmo padrão independentemente de sua posição (invariância à translação). Isso reduz drasticamente o número total de parâmetros do modelo e o torna mais eficiente.

Figura 18 – Esquema conceitual de redes neurais convolucionais.



Fonte: Próprio autor

Embora sejam famosas por tarefas de classificação, onde um *encoder* convolucional é seguido por camadas densas, as CNNs são também a base para tarefas de reconstrução de imagem. Para essas tarefas, como a redução de ruídos, a arquitetura mais comum é o *autoencoder* convolucional.

Conforme aplicado neste trabalho, esta arquitetura é composta por duas partes: um *encoder* (codificador) e um *decoder* (decodificador). O *encoder* aprende a comprimir a imagem em uma representação latente compacta, utilizando camadas convolucionais e de *pooling* (agrupamento) para extrair características e reduzir a dimensão espacial (*downsampling*). O *decoder* realiza o processo inverso: ele aprende a reconstruir a imagem original a partir da representação latente, tipicamente usando camadas de *upsampling* (ou *pooling* reverso) e convoluções (Goodfellow *et al.*, 2016).

Para a construção da arquitetura convolucional, diversos hiperparâmetros são inseridos, além daqueles apresentados anteriormente. Os mais relevantes para este projeto são:

- Filtros (*Filters*): Define o número de filtros aplicados nas camadas convolucionais. Cada filtro é responsável por detectar características específicas nas imagens, como bordas, texturas ou padrões. Um número maior de filtros permite que o modelo capture uma variedade mais ampla de características, aumentando sua capacidade de representação. No entanto, isso também implica em maior complexidade computacional e risco de *overfitting*.

- Tamanho do *Kernel* (*Kernel Size*): Determina a dimensão da janela que percorre a imagem durante a operação de convolução. *Kernels* menores (por exemplo, 3x3) capturam detalhes locais, enquanto *kernels* maiores (por exemplo, 5x5) abrangem áreas mais globais, identificando padrões mais globais. A escolha do tamanho do *kernel* afeta a capacidade do modelo de extrair informações relevantes das imagens.

- Tamanho do Pooling Reverso (*Up Pool Size*): No processo de decodificação do *autoencoder*, o *pooling* reverso (ou *upsampling*) é utilizado para aumentar a dimensão espacial das representações, reconstruindo a imagem original a partir das características extraídas. O tamanho do *pooling* reverso determina o fator pelo qual as dimensões são ampliadas. Um tamanho maior resulta em uma reconstrução mais detalhada, mas também pode introduzir artefatos se não for cuidadosamente ajustado.

3.3 Treinamento e Validação

O processo de treinamento de uma rede neural vai além da simples otimização de parâmetros. Envolve uma metodologia rigorosa para garantir que o modelo não apenas se ajuste aos dados de treinamento, mas que também generalize para novos dados. Esta seção descreve as etapas principais desse processo: a preparação dos dados, a estratégia de divisão do conjunto de dados, e as técnicas para diagnosticar e mitigar o superajuste (*overfitting*).

3.3.1 Preparação e Normalização dos Dados

A etapa inicial do treinamento é o pré-processamento dos dados. Redes neurais são sensíveis à escala das características de entrada, e características com magnitudes muito diferentes podem fazer com que o otimizador convirja lentamente ou de forma instável. Para solucionar isso, aplica-se a normalização.

No contexto de imagens, onde os valores dos *pixels* tipicamente variam no intervalo [0, 255], uma prática comum é reescalonar esses valores para o intervalo [0, 1]. Este passo assegura que todas as entradas contribuam de forma mais equilibrada para o processo de aprendizado e acelera a convergência do modelo (Goodfellow *et al.*, 2016).

$$h(x) = \frac{x}{255} \tag{17}$$

Onde h é o número o novo valor do *pixel*, x é o valor real do *pixel*.

3.3.2 Divisão em Treinamento, Validação e Teste

Para avaliar a capacidade de generalização de um modelo, sua habilidade de performar em dados nunca vistos, o conjunto de dados é particionado em três subconjuntos independentes (Goodfellow *et al.*, 2016):

- **Conjunto de Treinamento:** Corresponde à maior parte dos dados (ex: 70-80%) e é utilizado diretamente no processo de otimização. O modelo aprende os padrões, ajustando seus pesos e *biases*, com base nestas amostras.

- **Conjunto de Validação:** Uma porção menor dos dados (ex: 10-15%) que não é usada para o ajuste dos pesos. O modelo é avaliado neste conjunto ao final de cada época (ciclo de treinamento). Seu desempenho é usado para o ajuste de hiperparâmetros e para monitorar o *overfitting*.
- **Conjunto de Teste:** Um subconjunto (ex: 10-15%) completamente separado, que não participa de nenhuma etapa do treinamento ou ajuste. Ele é utilizado uma única vez, após o modelo estar totalmente treinado, para fornecer a métrica final e imparcial de seu desempenho.

3.3.3 *Overfitting*, *Underfitting* e Generalização

Ao avaliar um modelo, o objetivo principal é alcançar uma boa generalização (19). Contudo, ao monitorar o desempenho nos conjuntos de treinamento e validação, dois problemas comuns que impedem essa generalização podem ser diagnosticados (Goodfellow *et al.*, 2016):

- **Subajuste (*Underfitting*):** Ocorre quando o modelo é muito simples para capturar os padrões subjacentes dos dados. O modelo apresenta um desempenho ruim tanto no conjunto de treinamento quanto no de validação.
- **Superajuste (*Overfitting*):** Ocorre quando o modelo é excessivamente complexo e se ajusta excessivamente aos dados de treinamento, aprendendo padrões espúrios ou detalhes irrelevantes que não se aplicam a novos dados. O desempenho no conjunto de treinamento continua melhorando (erro diminui), mas o desempenho no conjunto de validação começa a piorar (erro aumenta). O modelo perde sua capacidade de generalização.
- **Generalização (*Balanced*)**: Ocorre quando o modelo atinge o equilíbrio ideal entre simplicidade e complexidade. O modelo é complexo o suficiente para capturar os padrões subjacentes dos dados, sem memorizar as particularidades específicas do conjunto de treinamento. Apresenta um baixo erro tanto no conjunto de treinamento quanto no de validação, indicando que o modelo generaliza bem para dados novos e nunca vistos.

Figura 19 – Ilustração visual dos conceitos de *underfitting*, *overfitting* e *balanced* (Generalização).



Fonte: Arya (2022)

3.4 Fundamentos da Camada de Mistura

A camada de mistura é um escoamento cisalhante livre canônico, formado quando duas correntes de fluido com velocidades distintas se encontram e fluem paralelamente. Esse fenômeno é governado, primariamente, pela instabilidade de Kelvin-Helmholtz, que surge na interface de cisalhamento devido à diferença de velocidade ($\Delta U = U_1 - U_2$). Esta instabilidade amplifica perturbações iniciais, levando à formação de grandes estruturas coerentes conhecidas como rolos de vorticidade (*rollers*).

À medida que o escoamento evolui espacialmente, estas estruturas interagem através do mecanismo de emparelhamento de vórtices (*vortex pairing*), resultando no crescimento da espessura da camada de mistura e na transferência de energia das grandes escalas para as menores (Pope, 2000).

3.4.1 Microescala de Taylor e o Número de Reynolds

Para caracterizar o regime de turbulência, utiliza-se o número de Reynolds. Enquanto o Reynolds global é baseado em dimensões macroscópicas (como a espessura da camada), o estudo da turbulência local frequentemente recorre ao número de Reynolds baseado na microescala de Taylor (Re_λ).

A turbulência pode ser visualizada como uma cascata de energia: a energia cinética é injetada nas grandes escalas (tamanho do vórtice principal) e transferida sucessivamente para escalas menores, até ser dissipada em calor pela viscosidade nas escalas de Kolmogorov (as menores escalas do escoamento).

A microescala de Taylor (λ) é uma escala de comprimento intermediária entre as grandes escalas e as escalas de dissipação. Ela está associada à taxa de deformação do fluido e aos gradientes de velocidade. O Re_λ é definido como:

$$Re_\lambda = \frac{u'\lambda}{\nu} \tag{18}$$

onde u' é a flutuação da velocidade (raiz quadrada média) e ν é a viscosidade cinemática. Um $Re_\lambda \approx 250$, como utilizado neste trabalho, indica um escoamento com uma faixa inercial bem desenvolvida, ou seja, existe uma separação clara entre as estruturas que contêm energia e as que dissipam energia. Para a rede neural, isso implica o desafio de reconstruir não apenas a geometria macroscópica dos vórtices, mas também a "textura"fina da turbulência.

3.4.2 Simulação Numérica Direta (DNS)

A base de dados utilizada neste estudo foi obtida através de uma Simulação Numérica Direta (DNS). Diferente de outras abordagens como RANS (*Reynolds-Averaged Navier-Stokes*) ou LES (*Large Eddy Simulation*), que modelam total ou parcialmente a turbulência, a DNS resolve as equações de Navier-Stokes completas, dependentes do tempo, sem o uso de modelos de turbulência (Moin; Mahesh, 1998).

Isso significa que a malha computacional em uma DNS deve ser fina o suficiente para resolver até a menor escala espacial do escoamento (a escala de Kolmogorov, η). Consequentemente, o custo computacional é extremamente elevado, limitando a DNS a números de Reynolds moderados e geometrias simples.

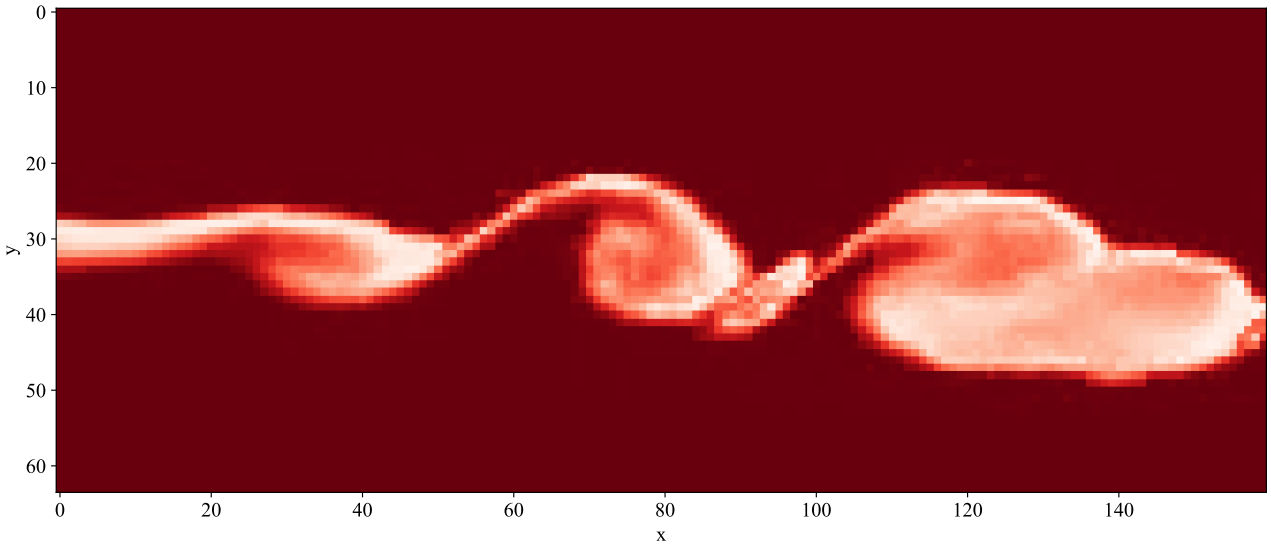
No contexto de filtragem de ruído com *Deep Learning*, o uso de dados de DNS é estratégico pois fornece o que se chama de *Ground Truth* (a verdade absoluta). Como a DNS resolve a física exata do escoamento, as imagens geradas são representações fiéis da realidade fluidinâmica, livres dos erros de modelagem presentes em outras técnicas numéricas, tornando-as ideais para o treinamento supervisionado do *autoencoder*.

4 METODOLOGIA

4.1 Base de dados

A base de dados aqui utilizada foi proveniente dos resultados de uma Simulação Numérica Direta (DNS) de uma camada de mistura com evolução espacial, a número de Reynolds baseado na microescala de Taylor, $Re_\lambda \approx 250$, do estudo conduzido por Fiscaletti *et al.* (2016). Ruído foi adicionado artificialmente às imagens originais obtidas da simulação. O conjunto de dados possui um total de 1.479 imagens, salvas no formato JPG com 160×64 *pixels* (20).

Figura 20 – Exemplo de uma imagem da simulação.

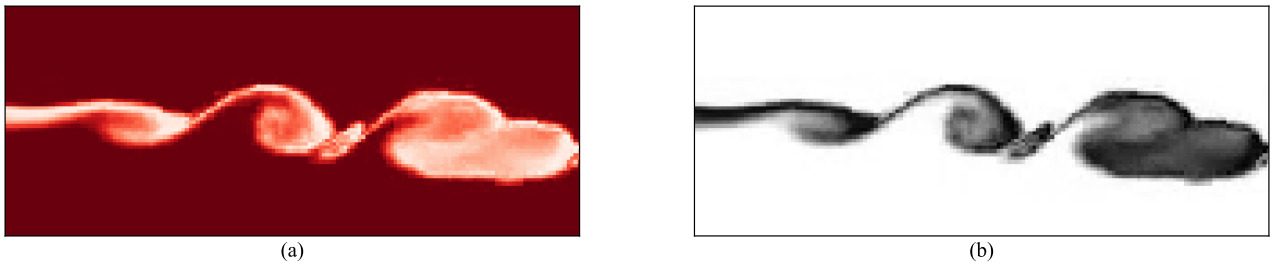


Fonte: Próprio autor

4.1.1 Pré-processamento de dados

As imagens foram convertidas para escala de cinza, reduzindo a dimensionalidade da entrada de três canais de cor RGB para um único canal (21). Essa simplificação diminui significativamente o custo computacional e a complexidade do modelo e acelera o tempo de treinamento.

Figura 21 – (a) Imagem Original, (b) Imagem com um único canal de cor.



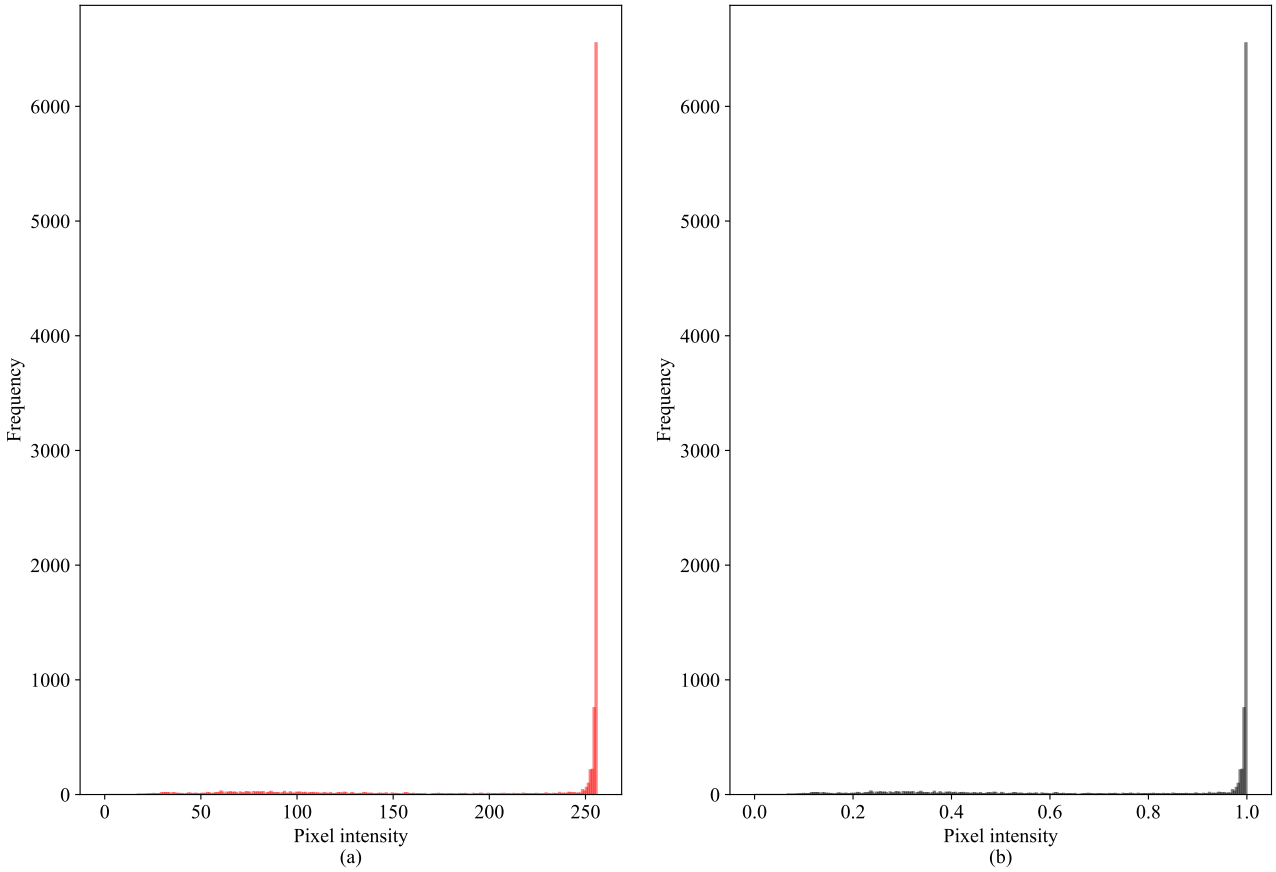
(a) (b)

Fonte: Próprio autor

Em seguida, os valores dos pixels foram normalizados de acordo com a equação 1 para o intervalo [0, 1] (ver 22 e 23). Garantindo que todas as características de entrada contribuam de forma equilibrada durante a atualização dos pesos da rede. A normalização acelera a convergência do algoritmo de otimização e evita a saturação das funções de ativação, resultando em um treinamento mais estável, rápido e em um modelo final mais robusto.

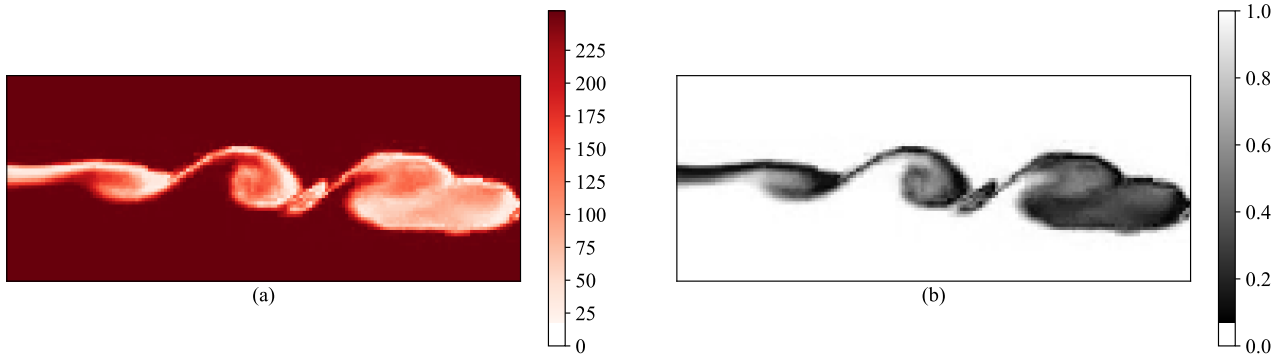
$$h(x) = \frac{x}{255} \tag{1}$$

Figura 22 – (a) Histograma da imagem original, (b) Histograma da imagem normalizada.



Fonte: Próprio autor

Figura 23 – (a) Imagem original, (b) Imagem normalizada.



Fonte: Próprio autor

4.1.2 Aumento de dados

A fim de aumentar a robustez e a capacidade de generalização do modelo, empregou-se a técnica de Aumento de Dados (*Data Augmentation*). A aplicação de transformações geométricas, como espelhamentos, serve a um propósito fundamental: ensinar à rede que o processo de remoção de ruído deve ser invariante à orientação do conteúdo da imagem.

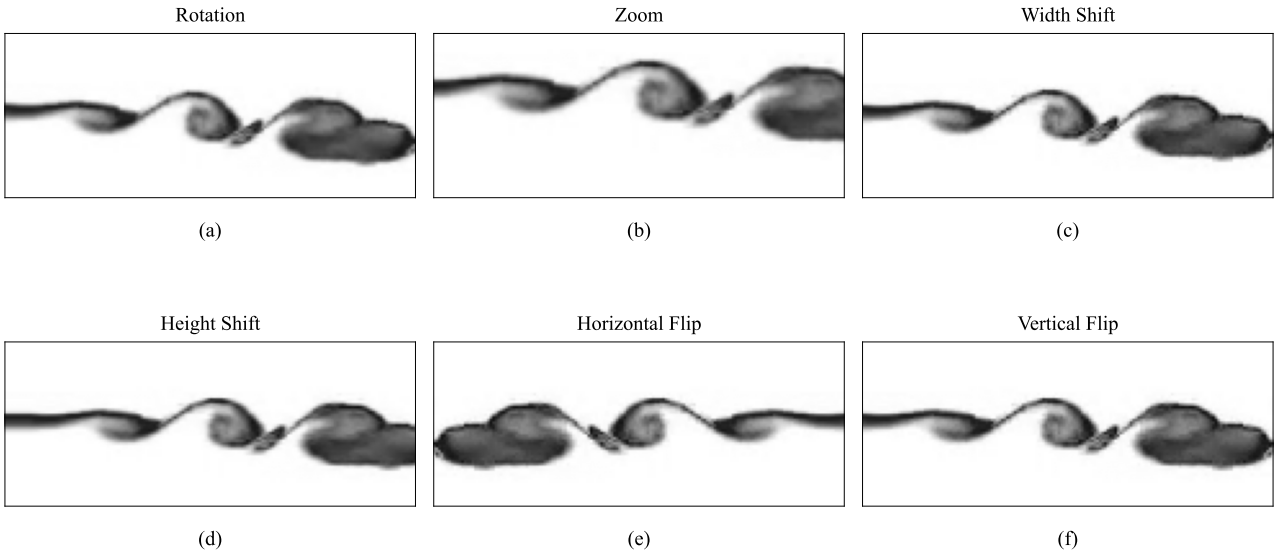
Ao expor a CNN a múltiplas versões da mesma imagem, o modelo é forçado a aprender uma representação mais fundamental das características do ruído e da estrutura da imagem. Este procedimento previne o *overfitting* ao conjunto de dados de treino e resulta em um modelo mais eficaz na remoção de ruído de imagens completamente novas e não vistas.

Para implementar essa estratégia, foram aplicadas transformações no conjunto de dados, aumentando artificialmente o número de amostras e totalizando 5 mil imagens. As operações de aumento de dados configuradas foram as seguintes:

- Espelhamento: Inversões aleatórias nos eixos horizontal e vertical.
- Rotação: Rotações aleatórias em um intervalo de até 10 graus.
- *Zoom*: Aproximações aleatórias de até 20% na imagem.
- Deslocamento: Translações horizontais e verticais de até 10% da dimensão da imagem.

A 24 apresenta um exemplo visual do resultado dessas operações.

Figura 24 – Exemplos de transformações de *data augmentation*: (a) Rotação, (b) Zoom, (c) Deslocamento de Largura, (d) Deslocamento de Altura, (e) Inversão Horizontal, e (f) Inversão Vertical.



Fonte: Próprio autor

4.1.3 Adição de Ruído

Para controlar a intensidade do ruído adicionado de forma padronizada e quantificável, foi utilizado o conceito de relação sinal-ruído (SNR). A SNR é uma medida adimensional que compara a potência

do sinal desejado (P_s) com a potência do ruído de fundo (P_N) (Johnson, 2006). Em sua forma linear, é definida como:

$$SNR = \frac{P_S}{P_N} \tag{2}$$

Na prática de engenharia e processamento de sinais, é comum expressar a SNR em escala logarítmica (decibéis - dB), dada pela equação:

$$SNR_{dB} = 10 \log_{10} \left(\frac{P_s}{P_N} \right) \tag{3}$$

Um valor de SNR alto (ex: $30dB$) indica que a potência do sinal é muito maior que a do ruído, resultando em uma imagem de alta fidelidade. Por outro lado, um SNR baixo (ex: $5dB$) indica que o ruído tem uma potência significativa em relação ao sinal, resultando em uma imagem muito degradada.

Neste trabalho, o processo foi o inverso da simples medição. Em vez de medir a SNR de uma imagem existente, um valor de SNR_{dB} alvo foi pré-estabelecido ($30dB$, $15dB$, $5dB$ e $0dB$). O objetivo era, então, calcular a potência do ruído (P_N) necessária para atingir essa SNR desejada, dada a potência da imagem original (P_s). O processo é o seguinte:

1. **Calcular a Potência do Sinal:** Para uma imagem, a potência do sinal (P_{sinal}) é aproximada pela variância (σ^2_{sinal}) dos valores de seus *pixels*.

2. **Converter a SNR de dB para linear:** A SNR_{dB} alvo é convertida de volta para a escala linear, invertendo a fórmula dos decibéis:

$$\text{SNR}_{\text{linear}} = 10^{\frac{\text{SNR}_{\text{dB}}}{10}}$$

3. **Determinar a Potência do Ruído:** Com a potência do sinal e a SNR linear desejada, podemos calcular a potência de ruído ($P_{\text{ruído}}$) necessária, rearranjando a fórmula original:

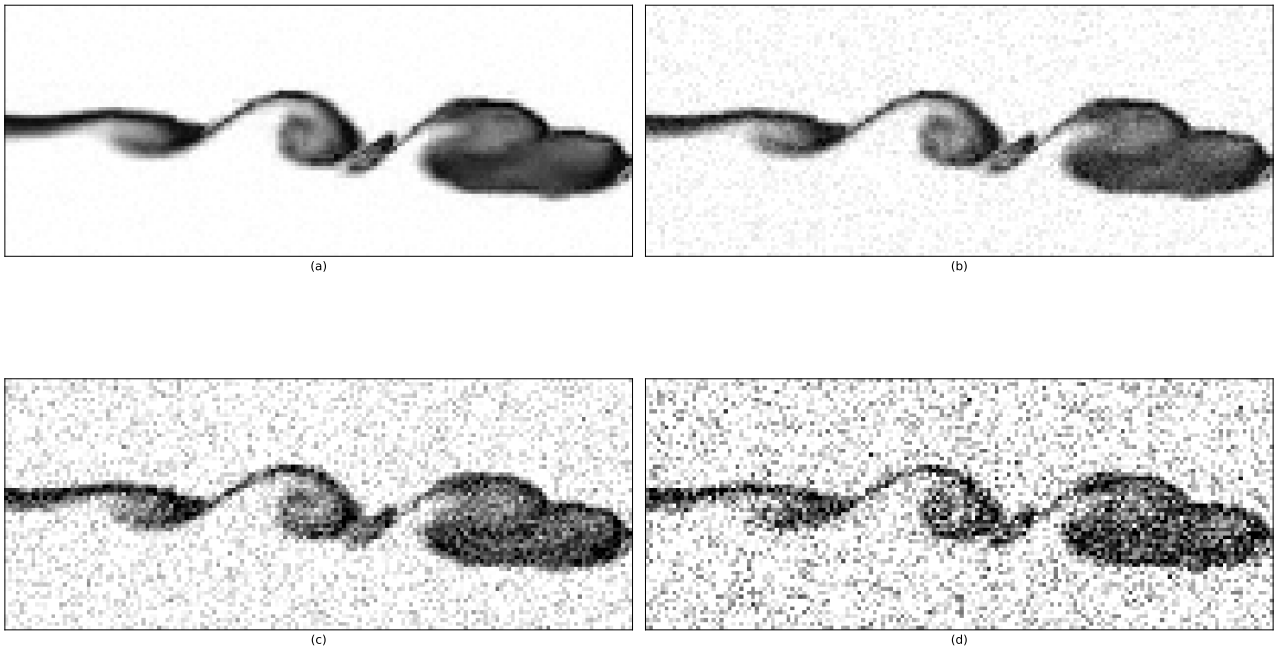
$$P_{\text{ruído}} = \frac{P_{\text{sinal}}}{\text{SNR}_{\text{linear}}}$$

4. **Gerar e Adicionar o Ruído:** Foi gerado um ruído aleatório de distribuição Gaussiana, conhecido como ruído branco aditivo gaussiano (AWGN - *Additive White Gaussian Noise*), com média zero e variância igual à $P_{\text{ruído}}$ calculada. Este ruído é então somado, *pixel* a *pixel*, à imagem original.

5. **Normalização (*Clipping*):** Os valores dos *pixels* da nova imagem ruidosa são truncados para garantir que permaneçam no intervalo válido (por exemplo, [0, 1] para imagens normalizadas).

Este método garante que o conjunto de dados de treinamento contenha exemplos com níveis de degradação consistentes e quantificáveis (25).

Figura 25 – a) Imagem com SNR $30dB$; b) Imagem com SNR$15dB$; c) Imagem com SNR $5dB$; d) Imagem com SNR $0dB$



Fonte: Próprio autor

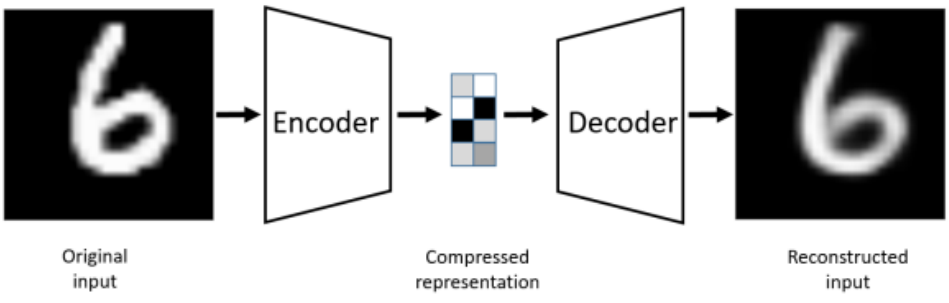
4.2 Arquitetura do *Autoencoder* Convolucional

Os *autoencoders* convolucionais são especialmente indicados para processamento de imagens, pois suas camadas convolucionais capturam relações espaciais locais em janelas receptivas, superando redes totalmente conectadas em tarefas de reconstrução. (Goodfellow *et al.*, 2016)

A estrutura central é o espaço latente — uma representação comprimida dos dados de entrada. O *encoder* mapeia a imagem de alta dimensionalidade para essa representação reduzida (camada *bottleneck* ou gargalo), forçando a rede a reter apenas características essenciais. O *decoder*, de forma espelhada, reconstrói a imagem original com máxima fidelidade a partir do espaço latente.

A arquitetura utilizada (26) emprega no *encoder* camadas convolucionais com *pooling* para redução dimensional e, no *decoder*, convoluções transpostas ou *upsampling*, garantindo saída nas mesmas dimensões da entrada.

Figura 26 – Exemplo de um *autoencoders* convolucionais.



Fonte: Bank, Koenigstein & Giryes (2023)

4.3 Divisão de Amostras de Treino e Teste

O conjunto de dados, composto por 5.000 imagens sintéticas com ruído gaussiano, foi dividido em:

- Treino: 70% (3.500 imagens)
- Validação: 10% (500 imagens)
- Teste: 20% (1.000 imagens)

Embora a divisão 80/20 seja amplamente adotada em aprendizado de máquina para promover generalização e evitar *overfitting* , a inclusão de um conjunto de validação explícito (10%) é justificada neste trabalho pela necessidade de monitoramento contínuo do erro de reconstrução durante o treinamento (Goodfellow *et al.*, 2016).

As imagens de entrada (*input*) são versões ruidosas dos sinais originais, enquanto as saídas esperadas (*output*) são as mesmas imagens de entrada — ou seja, o *autoencoder* é treinado de forma não supervisionada, com objetivo de reconstruir a entrada ruidosa a partir do espaço latente.

Essa abordagem é justificada porque, em cenários reais de simulação ou experimentação, não há acesso à versão limpa do sinal (sem ruído). Assim, o treinamento força o modelo a aprender representações robustas das características essenciais do sinal, eliminando o ruído como informação não essencial durante a compressão dimensional no *bottleneck*. O ruído, por ser aleatório e de alta entropia, tende a ser descartado na representação latente de baixa dimensionalidade, resultando em uma reconstrução mais limpa na saída.

4.4 Sintonização de Hiperparâmetros

O Optuna (Akiba *et al.*, 2019) é um *framework open-source* de otimização de hiperparâmetros baseado em técnicas bayesianas, projetado para ser flexível, eficiente e escalável. Diferentemente de métodos exaustivos como *grid search* ou aleatórios como *random search*, o Optuna utiliza o algoritmo *Tree-structured Parzen Estimator* (TPE) para modelar a relação entre hiperparâmetros e desempenho do modelo, priorizando configurações com maior potencial de sucesso com base em tentativas anteriores. Além disso, suporta *pruning* dinâmico que interrompe precocemente treinamentos promissores de forma a alocar recursos computacionais de maneira mais inteligente. Sua biblioteca permite definir espaços de busca complexos (inteiros, categóricos, contínuos em escala logarítmica) e integrar *callbacks* como *early stopping*, tornando-o especialmente adequado para tarefas de aprendizado profundo com alto custo computacional, como o treinamento de *autoencoders* convolucionais.

Foram realizados estudos independentes para cada nível de ruído (SNR), com até 100 épocas por estudo. A função objetivo minimiza o erro de validação após treinamento com *early stopping*.

Os hiperparâmetros otimizados estão listados na Tabela 1. Durante a construção do modelo, tentativas com *upsampling* incompatível com as dimensões de entrada (64×160) foram descartadas.

Tabela 1 – Hiperparâmetros otimizados pelo Optuna para cada nível de SNR.

Hiperparâmetro	**Tipo**	**Faixa/Valores**	**Descrição**
Número de Camadas	Inteiro	$[1, 4]$	Número de pares conv/pool no encoder
Filtros	Inteiro	$[4 : 4 : 256]$	Filtros por camada convolucional
Tamando do *Kernel*	Inteiro	$[3, 7]$	Tamanho do kernel convolucional
Função de Ativação	Categórico	13 opções[1]	Ativação nas camadas internas
Taxa de Aprendizado	Float (log)	$[10^{-6}, 10^{-4}]$	Taxa de aprendizado do otimizador Adam
Upsampling	Categórico	$\{2, 4, 8, 16\}$	Fator de *upsampling*
Tamanho do *Batch*	Categórico	$\{16, 32, 64, 128, 256\}$	Tamanho do lote de treinamento
Função de Ativação da Saída	Categórico	Mesmos parâmetros da Função de Ativação	Ativação na camada de saída
Loss	Categórico	{`mse`, `mae`, `mape`, `poisson`}	Função de custo

Fonte: Próprio autor

[1] tanh, relu, sigmoid, linear, leaky relu, softplus, softsign, selu, elu, exponential, swish, gelu.

5 RESULTADOS E DISCUSSÕES

5.1 Resultados da Sintonização de Hiperparâmetros

Nesta seção, são apresentados os resultados obtidos por meio do *framework* Optuna no processo de sintonização de hiperparâmetros.

Dada a natureza heterogênea dos dados resultantes da otimização, a visualização dos hiperparâmetros foi estratificada em três grupos distintos. Essa abordagem justifica-se pela necessidade de agrupar variáveis com propriedades matemáticas (escalas numéricas, logarítmicas ou categóricas) e semânticas semelhantes. A estratégia de agrupamento adotada seguiu os seguintes critérios:

- **Painel Estrutural (Parâmetros Físicos):** Agrupa variáveis inteiras que definem a topologia e a capacidade da rede neural: Número de camadas, tamanho do *kernel* e filtros.

  Como a quantidade de filtros (ordem de centenas) possui uma escala numérica muito superior ao número de camadas e tamanho de *kernel* (unidades), utilizou-se um gráfico com eixo Y secundário (*twin axis*) para permitir a correlação visual simultânea dessas grandezas sem perda de resolução.

- **Painel de Otimização (Dinâmica de Treinamento):** Reúne os hiperparâmetros que controlam a descida do gradiente e a convergência do modelo: Taxa de aprendizado e tamanho do *batch*.

  Devido à sensibilidade da taxa de aprendizado, que varia em ordens de grandeza (de 10^{-6} a 10^{-4}), a utilização de uma escala logarítmica no eixo Y é mandatória para a correta interpretação das flutuações. O tamanho do lote (*batch size*), sendo uma variável discreta, é plotado em conjunto para verificar correlações com a taxa de aprendizado.

- **Painel Qualitativo (Decisões de *Design*):** Composto pelas variáveis categóricas ou nominais: Função de ativação (interna e saída), função de custo (*loss*) e fator de *upsampling*.

  Por não se tratarem de valores progressivos contínuos, mas sim de escolhas de arquitetura, optou-se por gráficos de dispersão categórica (*categorical scatter plots*). Essa representação permite identificar visualmente se houve mudanças abruptas nas escolhas de *design*.

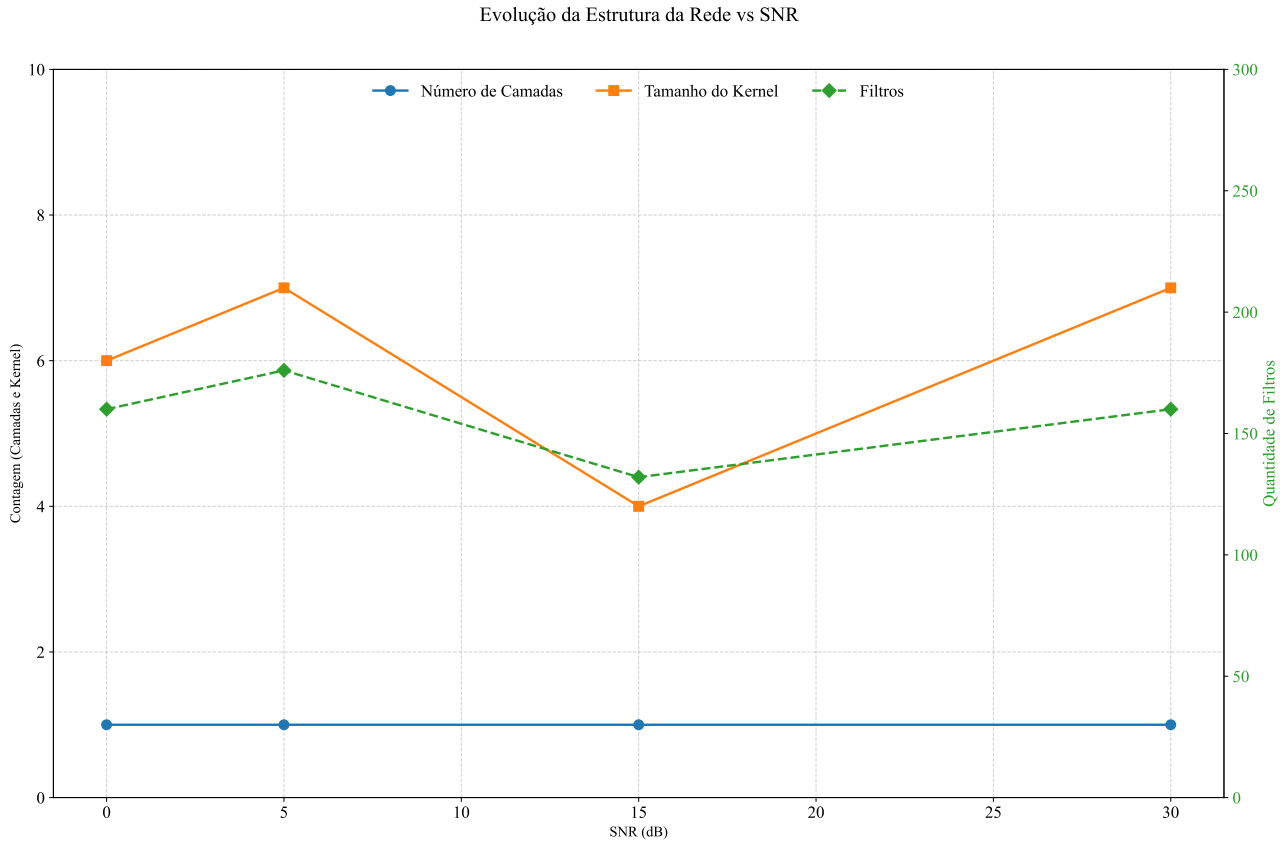
As seções a seguir detalham os resultados observados em cada um desses painéis.

5.1.1 Painel Estrutural - Topologia e Capacidade de Compressão

A 27 apresenta a evolução conjunta dos hiperparâmetros estruturais da rede neural convolucional — número de filtros, tamanho do *kernel* e número de camadas — em função da relação Sinal-Ruído (SNR).

Para permitir a visualização simultânea de grandezas com ordens de magnitude distintas, utilizou-se um gráfico com eixos verticais duplos: o eixo esquerdo (primário) representa contagens baixas (número de camadas e tamanho do *kernel*), enquanto o eixo direito (secundário, em verde) representa a quantidade de filtros.

Figura 27 – Evolução dos parâmetros estruturais (filtros, kernel e camadas) em relação à SNR.



Fonte: Próprio autor

Considerando que o treinamento foi realizado mapeando a entrada ruidosa para a própria saída ruidosa, a arquitetura encontrada pelo otimizador reflete a configuração ideal de "estrangulamento"(*bottleneck*) onde a rede prioriza a retenção de padrões estruturais (escoamento) em detrimento do ruído estocástico, que é dificilmente compressível. A análise detalhada revela:

- **Número de Camadas (Linha Azul contínua):** Diferentemente de abordagens supervisionadas clássicas, o otimizador convergiu para uma arquitetura rasa (1 camada) em todos os níveis de ruído testados (0 a 30 dB). Este comportamento sugere que, para a tarefa de auto-reconstrução sob a restrição do gargalo, o aumento da profundidade não trouxe ganhos significativos de abstração que justificassem o custo computacional ou o risco de *overfitting* ao ruído. A rede optou por realizar a filtragem através da manipulação espacial (*kernels*) e de capacidade (filtros) em uma única etapa de transformação não-linear.

- **Quantidade de Filtros (Linha Verde tracejada):** A quantidade de filtros, que dita a "largura"do gargalo de informação, variou entre aproximadamente 130 e 180.

  Em 0 dB e 5 dB (ruído intenso), o modelo demandou uma capacidade maior (160 a 176 filtros). Isso indica a necessidade de uma largura de banda maior para conseguir codificar as estruturas do fluido que estão sob o ruído intenso.

Em SNR $15\ dB$, houve uma redução na capacidade requerida (mínimo local de 132 filtros), sugerindo um ponto ótimo onde o sinal é claro o suficiente para ser comprimido eficientemente com menos parâmetros.

Em SNR $30\ dB$ (sinal quase limpo), a contagem volta a subir (160 filtros), provavelmente para preservar os micro-detalhes e altas frequências do escoamento turbulento que, na ausência de ruído, são interpretados como informação relevante a ser mantida.

- **Tamanho do *Kernel* (Linha Laranja contínua):** O tamanho da janela de convolução desempenhou um papel crucial na suavização espacial.

  Nos extremos (SNR $0/5\ dB$ e $30\ dB$), o otimizador preferiu *kernels* grandes (6 e 7). No caso ruidoso, janelas grandes atuam como filtros passa-baixa mais agressivos, ajudando a mediação do ruído vizinho. No caso limpo, permitem capturar correlações espaciais mais amplas das estruturas coerentes.

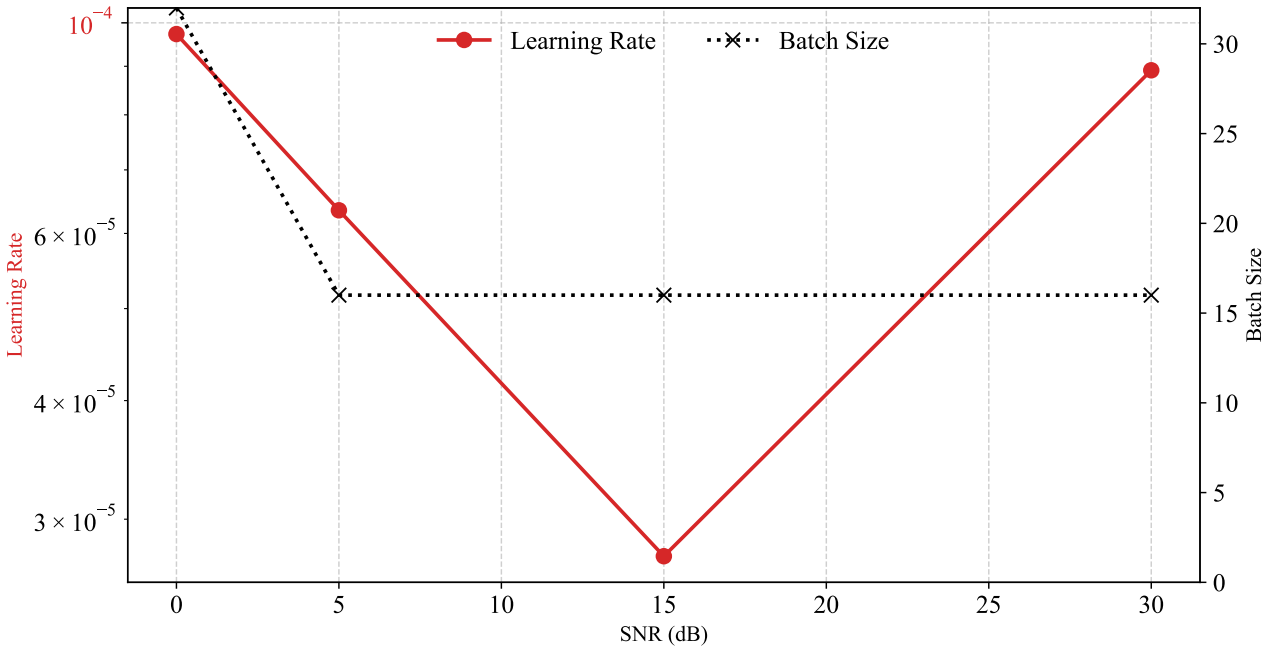
  Curiosamente, em SNR $15\ dB$, o tamanho do *kernel* caiu para 4. Coincidindo com a menor quantidade de filtros, este cenário parece representar um regime onde a filtragem local foi suficiente, sem necessidade de grande abrangência espacial ou alta capacidade de representação.

5.1.2 Painel de Otimização - Estabilidade e Convergência

A 28 detalha os hiperparâmetros que governam a dinâmica de atualização dos pesos da rede: a taxa de aprendizado (*learning rate*) e o tamanho do lote (*batch size*).

Para esta visualização, o eixo primário (esquerdo, em vermelho) exibe a taxa de aprendizado em escala logarítmica, permitindo observar as ordens de magnitude da descida do gradiente. O eixo secundário (direito, em preto) representa o tamanho do lote utilizado.

Figura 28 – Dinâmica de otimização (*learning rate* e *batch size*) em função da SNR.



Fonte: Próprio autor

A análise das curvas revela como o otimizador adaptou a estratégia de treinamento para lidar com a incerteza dos dados ruidosos na entrada e na saída:

- **Tamanho do Lote (Linha preta pontilhada):** Observa-se uma distinção clara baseada na severidade do ruído.

  Para SNR $0\ dB$, o otimizador selecionou um tamanho de lote maior (32). Como tanto a entrada quanto a saída contêm ruído severo, o gradiente calculado em poucas amostras seria extremamente instável (alta variância). Aumentar o lote para 32 permite calcular um gradiente médio mais representativo da física do problema, amortecendo as flutuações estocásticas do ruído.

  Para SNR $\geq 5\ dB$, o tamanho do lote convergiu para o mínimo (16). Com o sinal mais limpo, a estimativa do gradiente torna-se mais confiável, permitindo que o modelo se beneficie da regularização implícita de lotes menores para melhorar a generalização.

- **Taxa de Aprendizado (Linha vermelha contínua):** A taxa de aprendizado apresentou um comportamento não-monotônico, sugerindo diferentes regimes de navegação na superfície de erro:

  Em **SNR** $0\ dB$, notavelmente, a taxa foi a mais elevada ($\approx 10^{-4}$). Em uma superfície de custo dominada por ruído intenso (cheia de mínimos locais espúrios), uma taxa de aprendizado alta é necessária para fornecer "energia cinética" suficiente ao otimizador, permitindo que ele escape de poços de ruído e busque a estrutura global do escoamento.

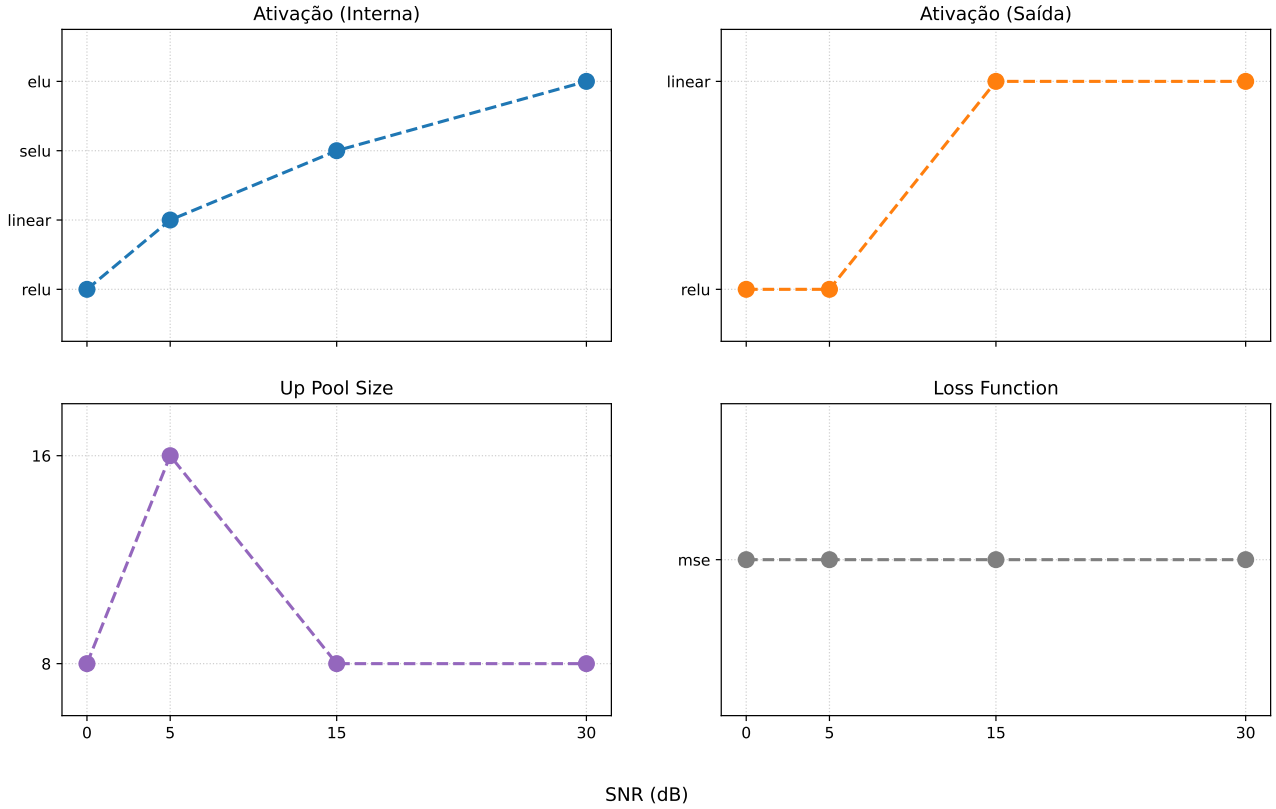
Em SNR $15\ dB$, encontrou-se o valor mínimo ($\approx 2,8 \times 10^{-5}$). Este ponto parece representar um regime de "ajuste fino", onde o sinal é claro o suficiente para não exigir grandes saltos, mas complexo o suficiente para demandar precisão na convergência.

Em SNR $30\ dB$, a taxa volta a subir ($\approx 9 \times 10^{-5}$), indicando que, na ausência de ruído significativo, a otimização pode proceder de forma agressiva e rápida em direção ao mínimo global.

5.1.3 Painel Qualitativo - Função de ativação (interna e saída), Função de custo (*loss*) e Fator de *upsampling*.

A 29 apresenta o painel qualitativo, agrupando as escolhas categóricas e discretas do otimizador: funções de ativação (interna e de saída), fator de *upsampling* e função de custo. Estas variáveis definem como a rede manipula a não-linearidade para realizar a filtragem via gargalo de informação.

Figura 29 – Seleção das funções de ativação, tamanho de *pooling* e função de custo em relação à SNR.



Fonte: Próprio autor

- **Ativação de Saída:**

  Para os cenários ruidosos (0 e $5\ dB$), o otimizador escolheu `relu`. Isso impõe uma restrição de não-negatividade na reconstrução final. Eliminando flutuações negativas de ruído que poderiam surgir no fundo da imagem, forçando uma limpeza agressiva.

  Para os cenários de sinal claro (15 e $30\ dB$), a escolha migrou para `linear`. A rede remove as restrições na saída, permitindo que o *autoencoder* reconstrua o campo de escoamento com total liberdade de amplitude.

- **Ativação Interna:**

  Em SNR $0\ dB$, o uso de `relu` favorece a esparsidade, ativando apenas os neurônios essenciais para representar as estruturas mais fortes e ignorando o ruído de fundo.

  Em SNR 15 e $30\ dB$, o otimizador adotou `selu` e `elu`. Estas são funções de ativação mais sofisticadas que preservam o fluxo de gradiente e permitem valores negativos internos.

- **Fator de *Upsampling* (*Up Pool Size*):**

  O comportamento predominante foi o uso de fator 8 (em 0, 15 e $30\ dB$). Para o ruído extremo ($0\ dB$), passos menores indicam cautela na reconstrução. Para sinais limpos (15 e $30\ dB$), o retorno ao fator 8 sugere uma preferência pela fidelidade espectral: evitar a interpolação excessiva (que ocorreria com fator 16) para preservar os gradientes finos dos vórtices.

  O caso de SNR $5\ dB$ foi o único a utilizar fator 16, indicando um ponto singular onde a relação sinal-ruído permitiu uma reconstrução espacial mais agressiva e suavizada.

- **Função de custo (*Loss Function*):** A manutenção do MSE em todos os casos reitera que, independentemente da estratégia de ativação, a minimização do erro quadrático continua sendo a métrica mais robusta para guiar o aprendizado.

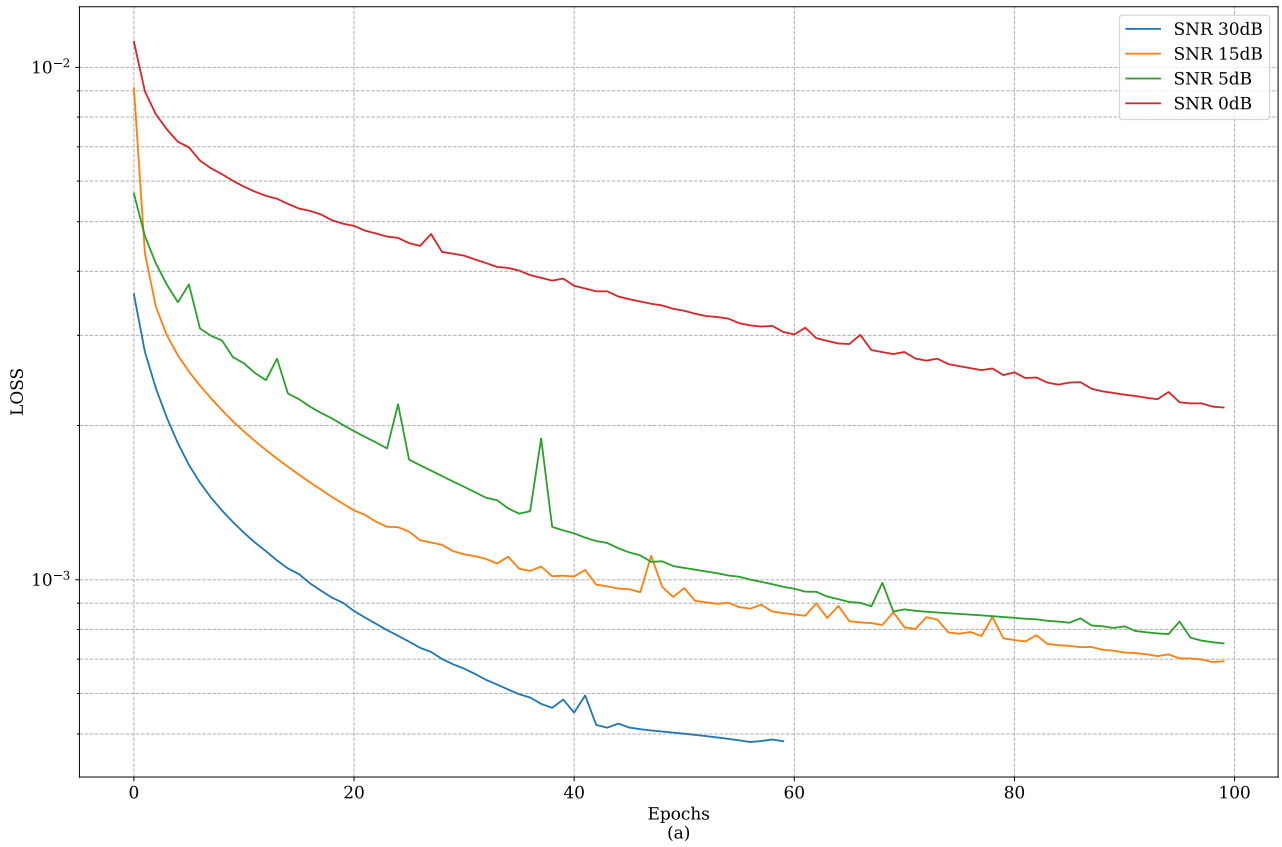
5.2 Desempenho do Treinamento e Análise de Convergência

A eficácia da arquitetura proposta em filtrar o ruído através do estrangulamento no espaço latente pode ser verificada diretamente pela análise das curvas de histórico de treinamento.

5.2.1 Convergência e Estabilidade por Nível de Ruído

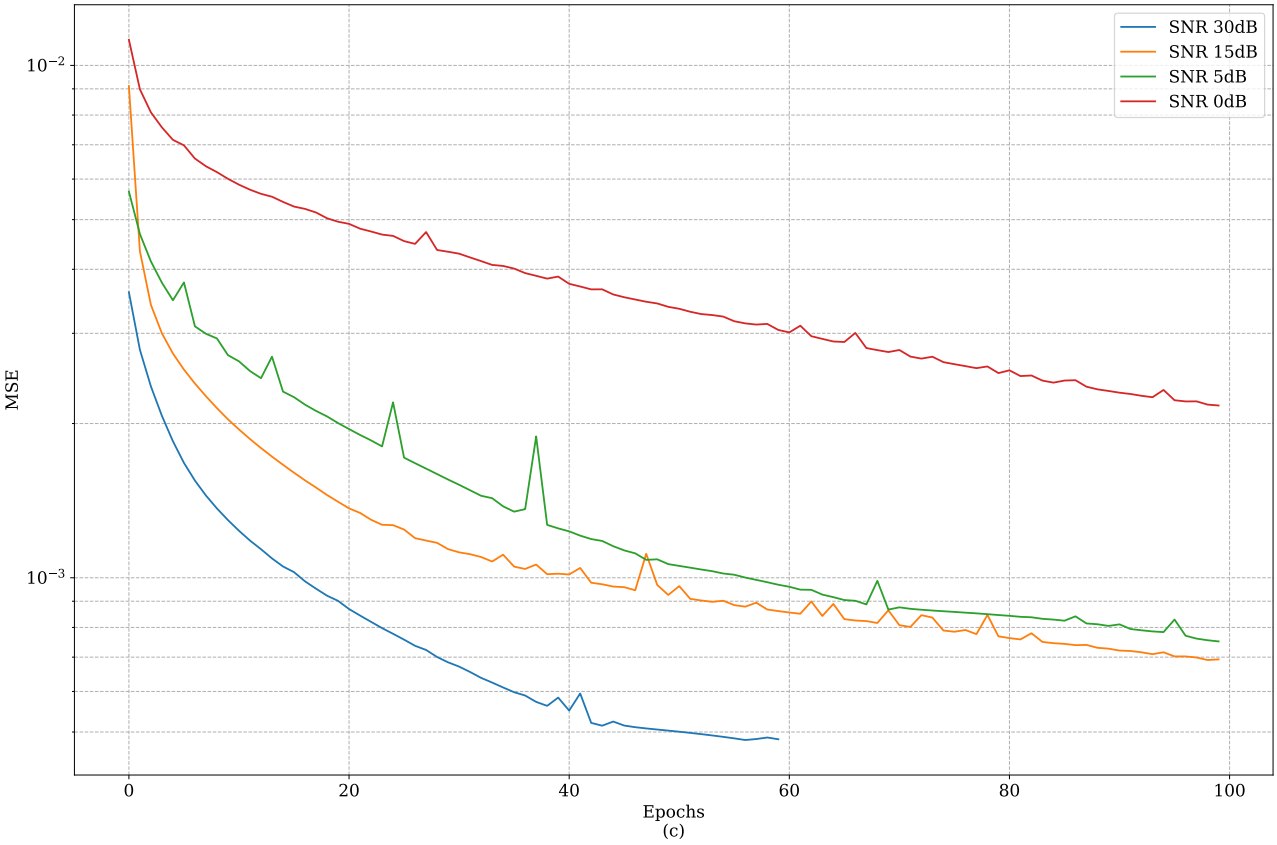
A eficácia do treinamento é evidenciada pela análise da função de custo (*loss*), bem como das métricas de erro quadrático médio (MSE) e erro absoluto médio (MAE). A 30 apresenta a evolução da função objetivo global, enquanto 31 e 32 detalham o comportamento específico das métricas de erro.

Figura 30 – Evolução da função de custo (*loss*) para os diferentes níveis de SNR.



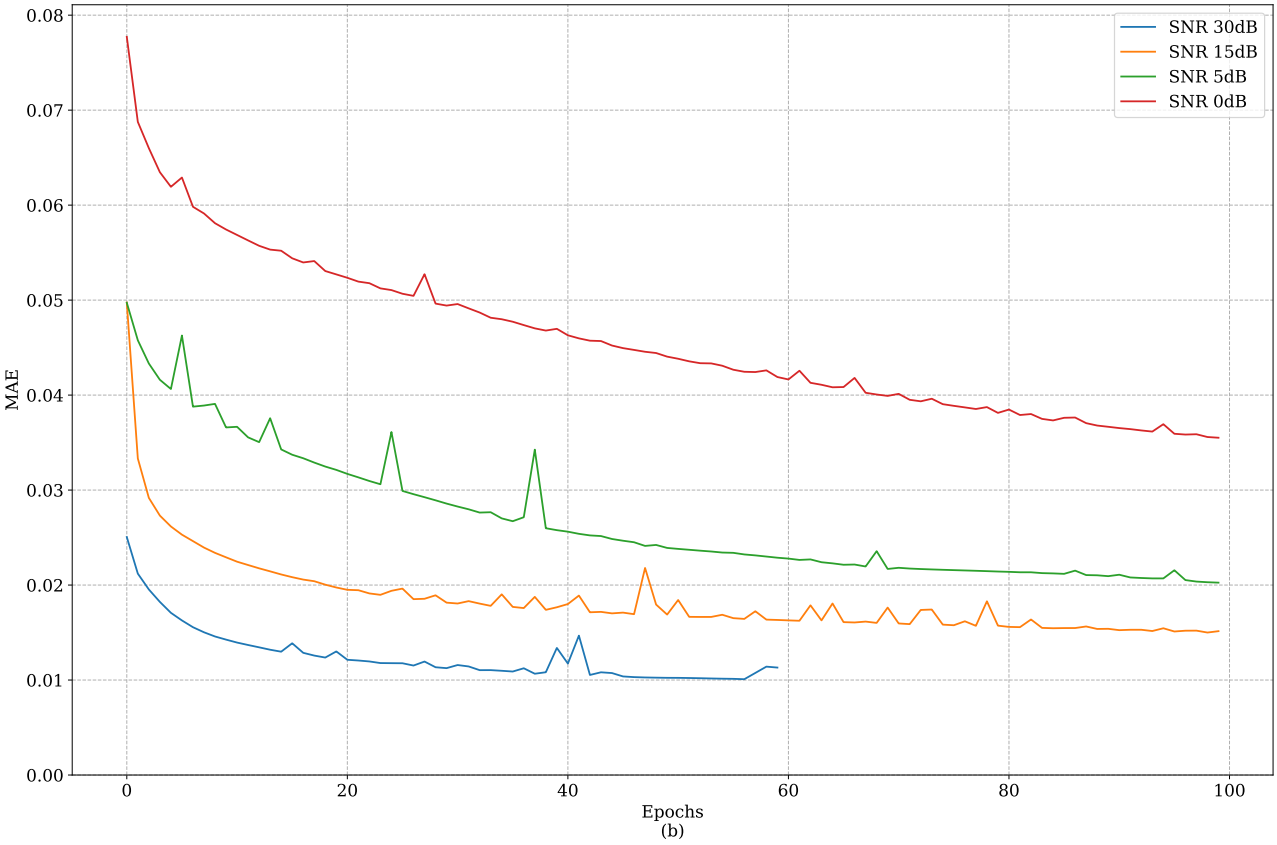
Fonte: Próprio autor

Figura 31 – Comparação das curvas de MSE durante o treinamento.



Fonte: Próprio autor

Figura 32 – Comparação das curvas de MAE durante o treinamento.



Fonte: Próprio autor

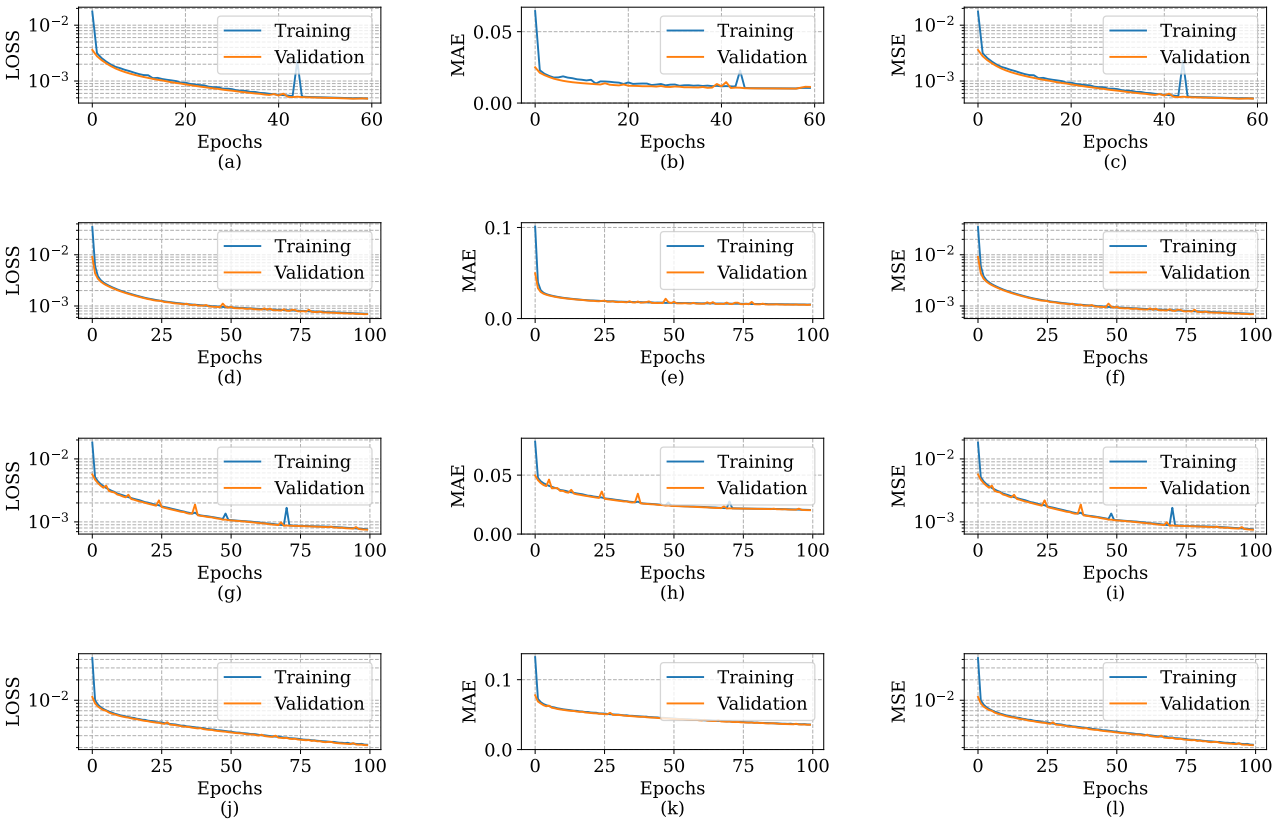
A análise conjunta destas curvas revela a dinâmica de otimização:

- **Minimização da Função Objetivo (*Loss*):** A curva de custo global confirma que o otimizador conseguiu navegar eficientemente na superfície de erro para todos os cenários. A consistência entre a curva de *Loss* e a curva de MSE indica que a otimização não sofreu distorções por termos de regularização excessivos, focando puramente na fidelidade de reconstrução.
- **Atuação do *Early Stopping*:** Observa-se que todas as arquiteturas atingiram um platô de estabilidade assintótica, validando a interrupção do treinamento pelo mecanismo de *Early Stopping*. Isso evitou o desperdício computacional e preveniu o *overfitting* (memorização do ruído).

5.2.2 Generalização e o Efeito do Gargalo

Para garantir que a redução de ruído não seja fruto de simples memorização (*overfitting*), analisa-se a discrepância entre os erros de treinamento e validação, conforme apresentado na 33.

Figura 33 – Curvas de custo de treinamento e validação para cada cenário de SNR.



Fonte: Próprio autor

A análise das curvas individuais revela que:

1. **Ausência de *Overfitting*:** Em todos os casos, a curva de validação (laranja) segue estritamente a tendência da curva de treinamento (azul), sem divergências significativas ao final das épocas.

Em suma, as curvas do treinamento definitivo validam a eficácia da seleção realizada pelo Optuna. Elas confirmam que os conjuntos de hiperparâmetros ótimos permitiram que a rede convergisse de forma robusta para cada cenário de ruído, minimizando o erro de reconstrução e assegurando a generalização para novos dados de escoamento, sem indícios de instabilidade ou *overfitting*.

5.3 Avaliação de Robustez e Generalização Cruzada

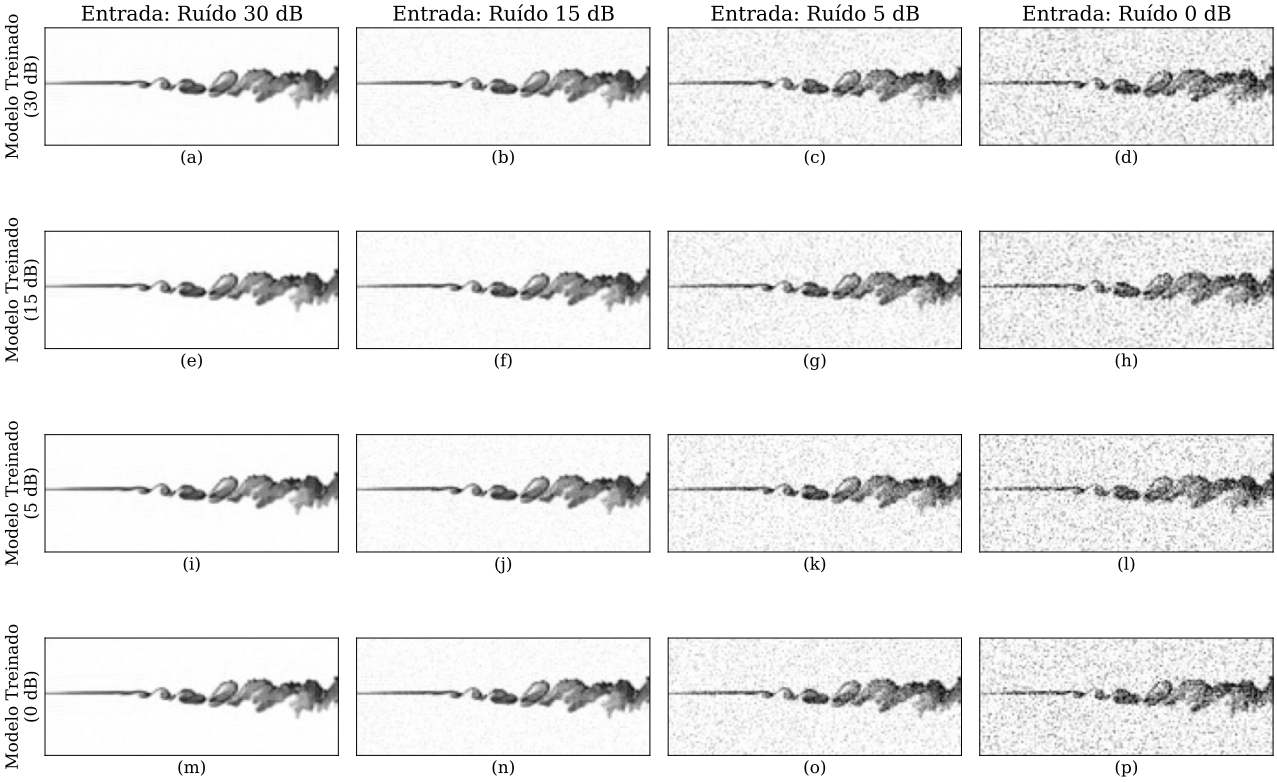
Após validar a convergência do treinamento, é crucial avaliar a robustez dos modelos gerados frente a cenários para os quais não foram especificamente otimizados. Esta etapa, denominada validação cruzada (*cross-validation*), consiste em submeter cada modelo treinado (Modelos $0\ dB$, $5\ dB$, $15\ dB$ e $30\ dB$) a todos os níveis de ruído do conjunto de teste.

O objetivo é investigar se a arquitetura otimizada para um gargalo específico mantém sua eficácia quando a intensidade do ruído muda, ou se ela degrada o sinal limpo.

5.3.1 Reconstrução Visual e Topologia do Escoamento

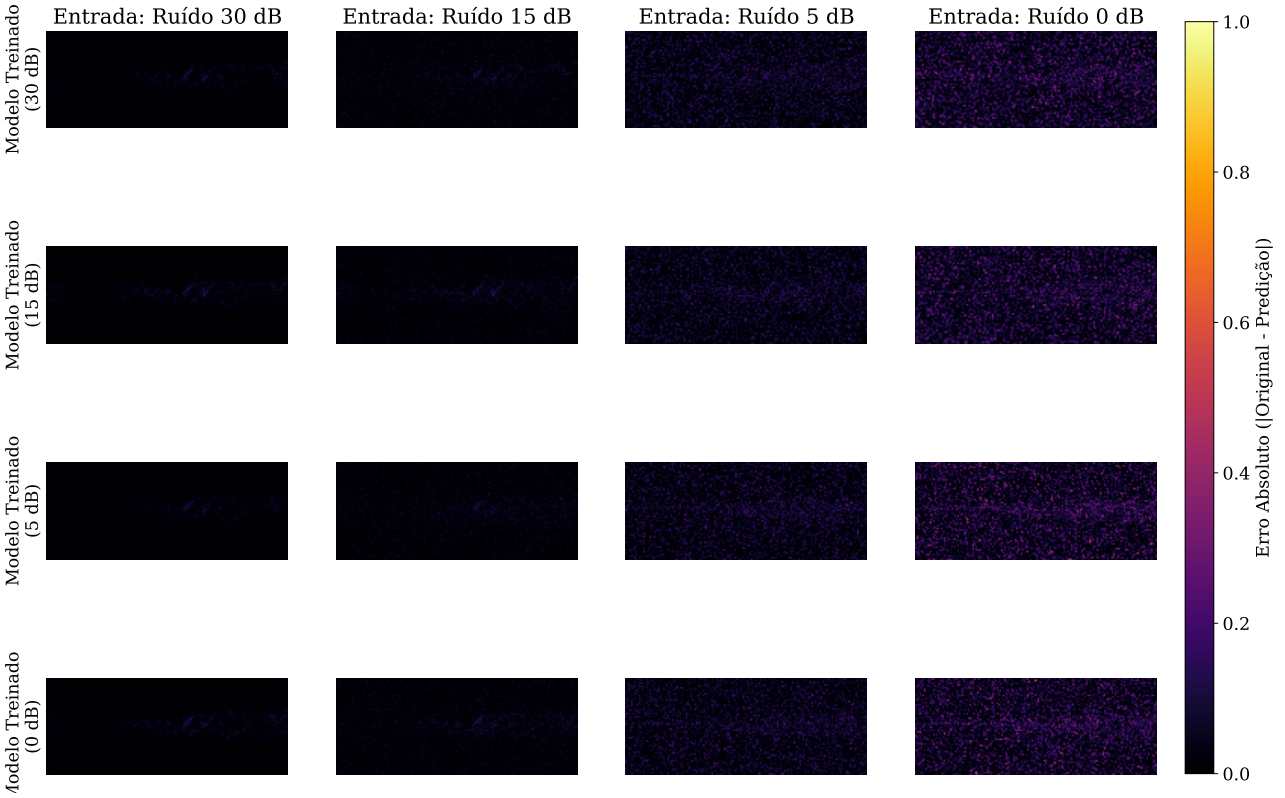
A matriz de reconstrução (34) permite avaliar a qualidade visual do escoamento recuperado, enquanto os mapas de calor (35) revelam a localização e a magnitude dos desvios, onde regiões escuras indicam alta fidelidade e regiões iluminadas apontam discrepâncias.

Figura 34 – Matriz de reconstrução do escoamento. As linhas indicam o modelo utilizado e as colunas o nível de ruído da imagem de entrada.



Fonte: Próprio autor

Figura 35 – Mapas de erro absoluto. A intensidade luminosa representa a magnitude da discrepância local entre a imagem original e a predita.



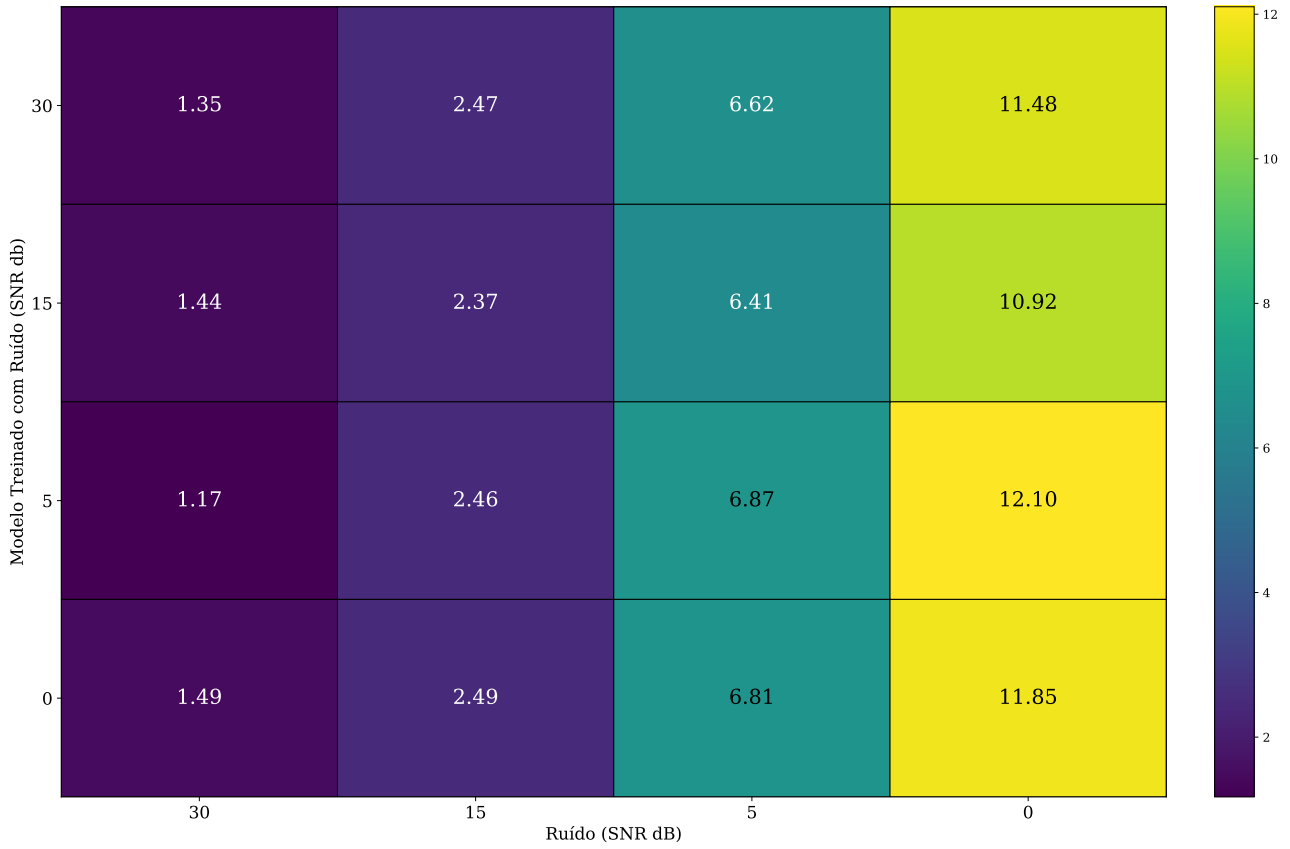
Fonte: Próprio autor

Observa-se que todos os modelos foram capazes de recuperar a topologia da esteira de vórtices (von Kármán) quando operando em suas condições de projeto (34-a, f, k, p). Os mapas de erro correspondentes na 35 apresentam predominância de tons escuros, validando a eficácia do *autoencoder* em extrair as características dominantes do escoamento com alta precisão quando o ruído de entrada corresponde ao do treinamento.

5.3.2 Análise Quantitativa do Erro de Reconstrução

Complementarmente à análise visual, a 36 sumariza o desempenho numérico em termos da raiz do erro quadrático Mmédio (RMSE). A matriz apresenta os valores de erro calculados para cada combinação de treinamento (linhas) e teste (colunas).

Figura 36 – Matriz de desempenho (RMSE) dos modelos treinados sob diferentes condições de ruído.



Fonte: Próprio autor

A análise da região triangular inferior da matriz de reconstrução (onde $SNR_{treino} < SNR_{teste}$) revela uma capacidade de generalização descendente. Observa-se que os modelos treinados em regimes de alta entropia (ex: $0\ dB$ e $5\ dB$) mantêm um desempenho satisfatório quando submetidos a cenários de ruído mais brando ($15\ dB$ e $30\ dB$).

6 CONCLUSÃO

Neste trabalho foi realizado o ajuste de hiperparâmetros de autoencoders convolucionais para remoção de ruído em imagens de simulações numéricas de escoamento em camada de mistura, utilizando uma base de dados composta por imagens com diferentes níveis de ruído artificialmente adicionados. Primeiramente, observou-se que o *framework* Optuna foi essencial para a otimização eficaz dos parâmetros da rede, permitindo adequar o modelo para diferentes níveis de ruído.

Utilizando as redes neurais otimizadas, verificou-se que o ruído foi removido para a maior parte dos níveis testados. Contudo, para níveis de ruído muito altos (exemplo: $0\ dB$), a remoção não foi totalmente eficaz. A análise da região triangular inferior da matriz de reconstrução, onde $SNR_{treino} < SNR_{teste}$, revelou uma capacidade de generalização descendente. Observou-se que modelos treinados em regimes de alta entropia (como $0\ dB$ e $5\ dB$) mantêm desempenho satisfatório quando aplicados a cenários com ruídos mais brandos ($15\ dB$ e $30\ dB$).

Esses resultados são particularmente relevantes para experimentos em escoamentos de fluidos, onde ruídos intensos comprometem a qualidade e interpretação dos dados. O *autoencoder* demonstrou ser uma ferramenta promissora para pré-processamento e remoção de ruído nestes contextos.

Para trabalhos futuros, sugere-se testar a metodologia em outras bases de dados e tipos de simulações, explorar diferentes funções de otimização e técnicas de regularização, além de avaliar a aplicação da métrica *structural similarity index* (SSIM) (Wang *et al.*, 2004) para uma análise mais qualitativa e perceptual da qualidade das imagens reconstruídas, complementando as métricas tradicionais e auxiliando na otimização do modelo para preservação de detalhes visuais essenciais.

REFERÊNCIAS

AKIBA, T. *et al.* Optuna: A next-generation hyperparameter optimization framework. *In*: **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [*S.l.*: *s.n.*], 2019.

ARYA, N. **How to Avoid Overfitting**. [*S.l.*]: KDnuggets, 2022. Disponível em: https://www.kdnuggets.com/2022/08/avoid-overfitting.html. Acesso em: 1 nov. 2025.

ASHGRIZ, N.; MOSTAGHIMI, J. An introduction to computational fluid dynamics. **Fluid flow handbook**, Citeseer, v. 1, p. 1–49, 2002.

BANK, D.; KOENIGSTEIN, N.; GIRYES, R. Autoencoders. **Machine learning for data science handbook: data mining and knowledge discovery handbook**, Springer, p. 353–374, 2023.

FISCALETTI, D. *et al.* Scale interactions in a mixing layer–the role of the large-scale gradients. **Journal of Fluid Mechanics**, Cambridge University Press, v. 791, p. 154–173, 2016.

GOODFELLOW, I. *et al.* **Deep learning**. [*S.l.*]: MIT press Cambridge, 2016. v. 1.

HAYKIN, S. **Neural networks: a comprehensive foundation**. [*S.l.*]: Prentice hall PTR, 1994.

JOHNSON, D. H. Signal-to-noise ratio. **Scholarpedia**, v. 1, n. 12, p. 2088, 2006.

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

MOIN, P.; MAHESH, K. Direct numerical simulation: a tool in turbulence research. **Annual review of fluid mechanics**, Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA, v. 30, n. 1, p. 539–578, 1998.

POPE, S. B. **Turbulent flows**. [*S.l.*]: Cambridge university press, 2000.

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

WANG, Z. *et al.* Image quality assessment: from error visibility to structural similarity. **IEEE transactions on image processing**, IEEE, v. 13, n. 4, p. 600–612, 2004.

WHITE, F. M. **Mecânica dos fluidos**. [*S.l.*]: McGraw Hill Brasil, 1988.