UNIVERSIDADE FEDERAL DO MARANHÃO

UNIVERSIDADE FEDERAL DO PIAUÍ
Doutorado em Ciência da Computação Associação UFMA/UFPI
Francisco dos Santos Viana

INSERÇÃO COLABORATIVA DE CAMADAS EM REDES *STACKED AUTOENCODER*

Orientador: Prof. Dr. Areolino de Almeida Neto

Co-orientador: Prof. Dr. Carlos M. M. de O. P. Soares

São Luís - MA
Março, 2026

Francisco dos Santos Viana

INSERÇÃO COLABORATIVA DE CAMADAS EM REDES *STACKED AUTOENCODER*

TESE DE DOUTORADO

Tese apresentada como requisito parcial para obtenção do título de Doutor em Ciência da Computação, ao Doutorado em Ciência da Computação, Associação UFMA/UFPI.

Orientador: Prof. Dr. Areolino de Almeida Neto
Co-orientador: Prof. Dr. Carlos M. M. de O. P. Soares

São Luís - MA
Março, 2026

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

dos Santos Viana, Francisco.
Inserção Colaborativa de Camadas em Redes Stacked Autoencoder / Francisco dos Santos Viana. - 2026.
99 p.

Coorientador(a) 1: Carlos Manuel Milheiro de Oliveira Pinto Soares.
Orientador(a): Areolino de Almeida Neto.
Tese (Doutorado) - Programa de Pós-graduação Doutorado em Ciência da Computação - Associação UFMA/UFPI, Universidade Federal do Maranhão, São Luís, Maranhão, 2026.

1. Inserção Colaborativa. 2. Redes Stacked Autoencoder. 3. Aprendizado Autocoordenado. I. de Almeida Neto, Areolino. II. Milheiro de Oliveira Pinto Soares, Carlos Manuel. III. Título.

Francisco dos Santos Viana

INSERÇÃO COLABORATIVA DE CAMADAS EM REDES *STACKED AUTOENCODER*

A presente Tese de Doutorado foi avaliada e aprovada por banca examinadora composta pelos seguintes membros:

Prof. Dr. Areolino de Almeida Neto
Orientador
Universidade Federal do Maranhão

Prof. Dr. Carlos Manuel Milheiro de Oliveira Pinto Soares
Co-orientador
Faculdade de Engenharia da Universidade do Porto

Prof. Dr. Alexandre César Muniz de Oliveira
Universidade Federal do Maranhão

Prof. Dr. Pedro de Alcântara dos Santos Neto
Universidade Federal do Piauí

Prof. Dr. Guilherme de Alencar Barreto
Universidade Federal do Ceará

Prof. Dr. Ricardo Bastos Cavalcante Prudencio
Universidade Federal de Pernambuco

Certificamos que esta é a versão original e final da Tese de Doutorado que foi julgada adequada para obtenção do título de Doutor em Ciência da Computação.

gov.br
Documento assinado digitalmente
AREOLINO DE ALMEIDA NETO
Data: 23/04/2026 12:26:55-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Areolino de Almeida Neto
Orientador

Prof. Dr. Anselmo Cardoso de Paiva
Coordenador

São Luís - MA, 27 de Março de 2026

Dedico este trabalho à memória da minha avó ***Raimunda****, lá na cidade de Caxias, que partiu muito cedo e que imaginava que seu primeiro neto seria apenas mais um trabalhador da roça. Ela ficaria surpresa e orgulhosa ao ver até onde o neto dela chegou. Levo comigo o amor, a força e o exemplo que deixou. Esta conquista também é sua.*

Agradecimentos

A Deus, por sempre me ajudar, por me capacitar, fortalecer-me e colocar pessoas boas no meu caminho — todas elas citadas aqui com muito carinho e gratidão.

Ao meu orientador, Professor Areolino, por todo apoio, conselhos, paciência e por sempre se manter ao meu lado. Em vários momentos, foi mais que um orientador — foi um verdadeiro amigo, sendo um exemplo de humanidade e dedicação.

Ao meu coorientador, Professor Carlos Soares, pela recepção na Faculdade de Engenharia do Porto e pela contribuição essencial à pesquisa. Sua orientação e confiança foram fundamentais para o amadurecimento deste trabalho.

À minha mãe Elma e ao meu irmão Adriano, por todo amor, paciência e por estarem sempre ao meu lado, mesmo à distância. Nenhuma conquista seria possível sem o apoio e o carinho de vocês.

Aos meus colegas de pós-graduação Marcelo, Rômulo e Bianca, pela amizade, companheirismo e pelas conversas que tornaram os dias mais leves.

À minha amiga Madah e ao meu amigo Felipe, pelo apoio constante, pelas palavras de incentivo e por acreditarem em mim mesmo nos momentos mais difíceis.

À família CETEC, representada por Elma, Creuma, Fátima e Madah, que sempre acompanhou minha trajetória com orgulho e torcida, fazendo parte de cada etapa dessa caminhada.

Aos colegas do meu departamento, representados pelo antigo chefe, Professor Airton, pela compreensão e por sempre tentar ajustar meus horários enquanto eu ainda cursava disciplinas e não estava afastado.

Aos colegas do IFMA, Professores Almir e Eveline, pelos momentos incríveis vividos durante o doutorado sanduíche — pelas boas risadas, passeios e pela amizade que levarei comigo.

À minha amiga de adolescência Flávia, que reencontrei em Portugal depois de tantos anos, e ao seu marido Bruno, pela receptividade, amizade e pelos momentos de alegria e nostalgia que tornaram aquele período inesquecível. Também à Dona Josefa e ao seu marido Alfredo (que infelizmente faleceu), que conheci de maneira totalmente inusitada — quando eu estava perdido em Portugal — e com quem acabei construindo uma amizade sincera, cheia de boas conversas e lembranças.

Ao meu vizinho Nataniel e à sua esposa Ana Rosa, por todo apoio, torcida e carinho quando estive longe da minha mãe.

Às minhas tias Celma, Jeane, Salete e à minha tia-vó Letícia, por todo amor, cuidado, torcida e por estarem sempre presentes cuidando de mim e da minha mãe com tanto carinho.

Em memória do meu padrasto, José Maria Santos, que sempre torceu por mim e que, tenho certeza, teria muito orgulho desta conquista.

À Universidade Federal do Maranhão (UFMA), ao Instituto Federal do Maranhão (IFMA), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e à Faculdade de Engenharia da Universidade do Porto (FEUP), pelo apoio institucional e pela oportunidade de realização do doutorado sanduíche, fundamental para o desenvolvimento desta pesquisa. À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) — Brasil, pelo apoio financeiro (Código de Financiamento 001).

Resumo

O presente trabalho propõe um método inovador para o crescimento estruturado de redes neurais profundas do tipo *stacked autoencoder*, com o objetivo de superar limitações de abordagens convencionais de expansão de redes. Redes neurais profundas têm-se mostrado altamente eficazes em tarefas de classificação, reconhecimento de padrões e extração automática de características, mas a definição de topologias adequadas continua sendo um desafio. Nos métodos tradicionais, a inserção de novas camadas escondidas durante o treinamento frequentemente resulta no aumento do erro de saída e na degradação do conhecimento previamente adquirido, tornando dispendiosa a duração do processo de expansão e dependente de ajustes manuais complexos. O método desenvolvido nesta pesquisa realiza a adição paralela de uma nova camada escondida e de uma nova camada de saída à camada de saída já existente. Diferentemente das abordagens convencionais, as camadas adicionadas formam um novo fluxo de dados até a saída da rede, produzindo um ramo auxiliar de processamento, onde essas novas camadas aprendem sem degradar o conhecimento já adquirido. Com isso, a inserção desenvolvida consegue colaborar com as camadas já existentes, desta maneira caracterizando o método como colaborativo. Após a inserção, realiza-se uma etapa de integração, em que o ramo auxiliar e a camada de saída original são combinados de maneira a consolidar o aprendizado das novas camadas, garantindo que o erro global da rede apresente comportamento decrescente e, ao mesmo tempo, mantendo a estabilidade do aprendizado. Ao término do processo, a camada de saída anterior torna-se parte da nova camada escondida, formando uma rede com uma única camada de saída. O método proposto foi avaliado em diferentes conjuntos de dados de classificação, demonstrando sua capacidade de preservar o aprendizado previamente consolidado, de reduzir o erro de saída de maneira consistente e de manter a estabilidade durante a expansão da rede. Os resultados indicam que o método oferece uma alternativa robusta para a expansão automática de redes neurais adaptativas, aumentando a eficiência no consumo de tempo do treinamento, reduzindo a necessidade de ajustes manuais e promovendo maior flexibilidade e confiabilidade em aplicações de classificação e reconhecimento de padrões.

Palavras-chave: Inserção colaborativa, Redes *Stacked Autoencoder*, Aprendizado Autocoordenado.

Abstract

This work proposes an innovative method for the structured growth of deep neural networks of the stacked autoencoder type, aiming to overcome limitations of conventional network expansion approaches. Deep neural networks have proven highly effective in classification, pattern recognition, and automatic feature extraction tasks; however, defining appropriate topologies remains a challenge. In traditional methods, inserting new hidden layers during training often leads to an increase in output error and degradation of previously acquired knowledge, making the expansion process time-consuming and dependent on complex manual adjustments. The method developed in this research performs the parallel addition of a new hidden layer and a new output layer alongside the existing output layer. Unlike conventional approaches, the added layers create a new data flow toward the network output, forming an auxiliary processing branch in which the new layers learn without degrading previously acquired knowledge. As a result, the proposed insertion strategy collaborates with the existing layers, characterizing the method as collaborative. After insertion, an integration step is carried out in which the auxiliary branch and the original output layer are combined to consolidate the learning of the new layers. This ensures that the global network error exhibits a decreasing behavior while maintaining learning stability. At the end of the process, the previous output layer becomes part of the new hidden layer, forming a network with a single output layer. The proposed method was evaluated on different classification datasets, demonstrating its ability to preserve previously consolidated knowledge, consistently reduce output error, and maintain stability during network expansion. The results indicate that the method provides a robust alternative for the automatic expansion of adaptive neural networks, improving training time efficiency, reducing the need for manual adjustments, and promoting greater flexibility and reliability in classification and pattern recognition applications.

Keywords: Collaborative Insertion, Stacked Autoencoder Networks, Self-Coordinated Learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ALT	Inserções do tipo aleatória
BP	Algoritmo de retropropagação do erro (*Backpropagation*, do inglês)
CAE	Autoencoder contrativo (*Contractive Autoencoder*, do inglês)
CNN	Rede neural convolucional (*Convolutional Neural Network*, do inglês)
CNT	Inserções do tipo constante
CRT	Inserções do tipo crescente
DAM	Estimativa adaptativa de momentos (*Adaptive Moment Estimation*, do inglês)
DBN	Rede de crença profunda (*Deep Belief Network*, do inglês)
DRT	Inserções do tipo decrescente
FN	Falso negativo
FP	Falso positivo
ML	Aprendizado de máquina (*Machine Learning*, do inglês)
MLP	Perceptron multicamadas (*Multilayer Perceptron*, do inglês)
MSE	Erro quadrático médio (*Mean Squared Error*, do inglês)
RMSProp	Propagação da média quadrática (*Root Mean Square Propagation*, do inglês)
RNA	Rede neural artificial
SAE	Autoencoder empilhado (*Stacked Autoencoder*, do inglês)
SGD	Descida do gradiente estocástico (*Stochastic Gradient Descent*, do inglês)
VN	Verdadeiro negativo
VP	Verdadeiro positivo

Lista de símbolos

b_k	*bias* do neurônio
$f(.)$	função de ativação do neurônio
u	vetor de entradas do neurônio artificial
$g(\vec{u})$	função de combinação linear das entradas ponderadas
y	saída do neurônio da camada de saída
w	peso da conexão entre neurônios
x	entrada original de um neurônio
$\sum$	somatório
H	representação da saída da camada oculta de uma rede neural
O	representação da camada de saída
E	representação de camada extra no método de inserção colaborativa
I	matriz identidade
k	peso da conexão entre as camadas de saída de cada ramo
θ	função custo

Sumário

1 Introdução

As redes neurais artificiais (RNA) são empregadas em classificação, regressão, reconhecimento, controle e previsão, entre outras, graças à capacidade de aproximar mapeamentos não lineares complexos (HORNIK *et al.*, 1989; HAYKIN, 2009). O desempenho final, porém, depende fortemente da arquitetura, que define o quanto a rede consegue extrair padrões relevantes. Esse ponto tem sido enfatizado em estudos recentes sobre generalização, que mostram que a robustez do modelo está ligada à adequação estrutural da rede ao problema (JIANG *et al.*, 2020; KAPLUN *et al.*, 2022; ZHANG *et al.*, 2021).

Contudo, não existe um método sistemático para definir a topologia ideal. A escolha do número de camadas, de neurônios e funções de ativação ainda é realizada arbitrariamente, mesmo em arquiteturas amplamente estudadas como o Perceptron Multicamadas (MLP). Redes pequenas não capturam padrões essenciais; redes grandes sofrem sobreajuste e desperdiçam recursos (DANTAS, 2017). Além disso, o espaço de combinações cresce rapidamente, fazendo da busca por uma arquitetura adequada um problema computacionalmente caro, tema recorrente nas discussões sobre otimização estrutural em aprendizado profundo (WISTUBA, 2023). Tentativas mais modernas, como métodos de *neural architecture search* (NAS), amenizam parte desse problema, mas ainda apresentam alto custo e instabilidade (WHITE *et al.*, 2021; ELSKEN *et al.*, 2022).

A necessidade de representar padrões mais complexos levou à adoção de arquiteturas profundas. Entre elas, os *stacked autoencoders* (SAE) destacam-se pela capacidade de extrair representações progressivamente mais abstratas, obtidas por meio do empilhamento de autoencoders treinados de forma não supervisionada, seguido de um ajuste fino supervisionado, o que favorece melhor generalização e redução da dimensionalidade dos dados (VINCENT *et al.*, 2010). Estudos recentes mostram que a eficácia dessas arquiteturas depende fortemente da profundidade e da largura das camadas, reforçando a limitação de definir essas escolhas antecipadamente (BENGIO; COURVILLE, 2021; DONG; AL., 2023). Métodos adaptativos vêm sendo propostos para ajustar automaticamente a estrutura dos autoencoders, adicionando camadas ou neurônios conforme necessário (CHEN; AL., 2024; NI; AL., 2024). Essa tendência reforça que arquiteturas fixas não são ideais quando a complexidade dos dados varia de forma significativa.

Nesse cenário, métodos construtivos oferecem uma alternativa direta ao modelo convencional de definir primeiro e treinar depois. Em vez de projetar toda a topologia, a rede cresce de forma incremental durante o treinamento, adicionando novos neurônios e camadas apenas quando a estrutura atual deixa de ser suficiente (FAHLMAN; LEBIERE, 1990; LIN *et al.*, 2016). Avanços recentes mostram que abordagens incrementais conseguem melhorar

generalização, reduzir sobreajuste e adaptar a arquitetura à dificuldade real do problema (SHARIR; AL., 2023; YUAN; AL., 2023). Há também um movimento crescente em direção a arquiteturas dinâmicas, capazes de expandir seletivamente partes da rede, combinando princípios construtivos com busca estrutural eficiente (HEUILLET; BELAID, 2023; GAMBELLA; AL., 2025; WANG; AL., 2024).

Quando aplicados a SAE, métodos construtivos permitem ajustar automaticamente profundidade e largura, ampliando a capacidade de representação sem depender de arquiteturas pré-definidas. Trabalhos recentes mostram que esse crescimento controlado favorece a extração de características latentes e melhora a generalização (YAN; AL., 2021; NI; AL., 2024; PEDRAZA; AL., 2024). Além disso, mecanismos de expansão incremental vêm sendo integrados com aprendizado contínuo, permitindo que a rede evolua de acordo com novas informações (YE; AL., 2023; HAN; AL., 2023).

Diante de todo o exposto, este trabalho propõe um método colaborativo de construção incremental aplicado a stacked autoencoders. O objetivo é ajustar automaticamente profundidade e largura ao longo do treinamento, reduzindo a necessidade de intervenção manual, melhorando a generalização e aumentando a robustez do modelo frente a diferentes conjuntos de dados. O restante do texto apresenta os fundamentos teóricos, detalha o método proposto, descreve os experimentos e discute os resultados. O foco é contribuir para o avanço de arquiteturas profundas adaptativas, alinhadas às demandas atuais de flexibilidade e eficiência.

1.1 Motivação e justificativas

O processo de treinamento de redes neurais com uma topologia previamente definida é, frequentemente, uma tarefa complexa e demorada. A determinação do número ideal de camadas ocultas e de neurônios em cada uma delas exige um processo de experimentação intensivo, envolvendo diversas combinações e múltiplas execuções de treinamento. Além disso, topologias superdimensionadas podem levar a um aumento do custo computacional e à perda de capacidade de generalização, enquanto topologias subdimensionadas comprometem o desempenho do modelo.

Nesse contexto, métodos construtivos surgem como alternativas eficientes, permitindo a criação progressiva da rede a partir de uma topologia mínima. Tais métodos acrescentam novas camadas ou neurônios de forma controlada, guiados por critérios de desempenho e erro residual. Essa abordagem possibilita que o modelo evolua de modo adaptativo, ajustando sua complexidade apenas quando necessário para melhorar o aprendizado.

Entre essas estratégias, os autoencoders empilhados (*stacked* autoencoders) destacam-se por sua capacidade de extrair representações hierárquicas e não supervisionadas dos dados. Entretanto, a definição estática de sua arquitetura ainda representa um desafio, pois depende de conhecimento prévio sobre a estrutura mais adequada ao problema. Diante disso, o

método colaborativo construtivo proposto nesta pesquisa busca integrar a eficiência dos autoencoders empilhados com um mecanismo de crescimento dinâmico e coordenado entre camadas, permitindo o aprendizado incremental, a redução do esforço de configuração manual e a obtenção de modelos mais compactos e generalistas.

Essa abordagem visa, portanto, automatizar o processo de expansão estrutural da rede, minimizando o tempo de desenvolvimento e garantindo uma melhor adaptação do modelo a diferentes bases de dados e tarefas. Assim, o método contribui tanto para a eficiência computacional quanto para a robustez e flexibilidade das redes neurais profundas em contextos de aprendizado contínuo e construtivo.

Por fim, essa motivação culmina no desenvolvimento do modelo *Stacked Autoencoder Collaborative Network* (*SAE CollabNet*), proposto nesta tese, que busca combinar os princípios de aprendizado colaborativo e construtivo para gerar arquiteturas de redes neurais adaptativas, com desempenho competitivo e menor complexidade estrutural.

1.2 Objetivos

O objetivo geral desta pesquisa é propor, desenvolver e validar um método construtivo colaborativo baseado em autoencoders empilhados, denominado *SAE CollabNet*, capaz de construir redes neurais profundas de forma progressiva, a partir de uma topologia mínima, ajustando automaticamente sua estrutura de acordo com o desempenho obtido em cada etapa de aprendizado.

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

- Desenvolver um mecanismo de crescimento dinâmico de camadas ocultas, permitindo a expansão da rede conforme a necessidade de aprendizado e melhoria do erro residual;
- Implementar uma estratégia de colaboração entre ramos e camadas, de modo que o conhecimento adquirido por cada nova camada seja incorporado e compartilhado com as demais partes da rede;
- Avaliar diferentes estratégias de crescimento (constante, crescente, decrescente e aleatória) para determinar a mais adequada à convergência e à generalização do modelo;
- Integrar mecanismos de avaliação do desempenho parcial da rede para controlar o processo de adição de camadas e evitar sobreajuste (*overfitting*);
- Realizar experimentos comparativos utilizando diferentes conjuntos de dados e métricas de desempenho, a fim de validar a eficácia e a estabilidade do método proposto;
- Analisar a influência de funções de ativação e ajustes colaborativos de pesos sobre o desempenho global da rede;

- Desenvolver uma interface de controle e visualização dos resultados, facilitando a configuração dos parâmetros e o acompanhamento do processo de crescimento da rede.

1.3 Trabalhos Relacionados

Os estudos em *deep learning* têm evoluído rapidamente, impulsionados pela demanda crescente por modelos capazes de extrair representações cada vez mais profundas e generalizáveis dos dados. O avanço das arquiteturas, das técnicas de regularização e dos algoritmos de otimização tem possibilitado o desenvolvimento de redes neurais de grande escala, capazes de aprender padrões complexos em domínios como visão computacional, processamento de linguagem natural e sistemas de recomendação. Contudo, o processo de definição de uma topologia adequada e o treinamento eficiente de modelos profundos ainda constituem desafios abertos, principalmente no que refere-se à escalabilidade e à estabilidade do aprendizado. Nesta seção, apresenta-se um panorama de trabalhos relevantes que exploram diferentes estratégias de crescimento, adaptação e otimização arquitetural de redes neurais profundas, com ênfase em abordagens relacionadas ao aprendizado incremental e construtivo.

Vincent *et al.* (2010) introduziram o conceito de *stacked denoising autoencoders*, estabelecendo um marco na construção hierárquica de representações. Cada camada é treinada para reconstruir uma versão corrompida da entrada, o que força o modelo a aprender características invariantes e robustas ao ruído. Esse paradigma evidenciou o potencial do treinamento não supervisionado em redes profundas, ao mitigar a dependência de grandes conjuntos de dados rotulados e promovendo inicializações mais estáveis para o aprendizado supervisionado subsequente.

Seguindo essa linha, Makhzani e Frey (2013) propuseram o *k-sparse autoencoder*, uma variação eficiente que restringe a ativação a apenas k unidades ocultas por amostra. Enquanto o *denoising* força a robustez frente a perturbações na entrada, essa restrição de esparsidade promove uma codificação mais parcimoniosa, favorecendo representações interpretáveis e generalizáveis. O modelo combina simplicidade computacional com bom desempenho em cenários de alta dimensionalidade, destacando-se por sua aplicabilidade em grandes volumes de dados.

Com o objetivo de tornar o treinamento de arquiteturas profundas mais estável, Kalmanovich e Chechik (2014) apresentaram o treinamento progressivo de autoencoders, em que novas camadas são inseridas de maneira incremental. Tal abordagem permite que a rede consolide gradualmente representações intermediárias antes de aumentar sua profundidade, reduzindo a instabilidade causada pela otimização simultânea de um grande número de parâmetros. Além disso, o método proposto obteve uma melhoria pequena, mas consistente, no erro de reconstrução e no erro de classificação no conjunto de dados de tamanho médio. Contudo,

como as camadas previamente treinadas ainda podem sofrer interferência durante ajustes globais, a degradação do erro não é completamente eliminada. Nesse contexto, estratégias que isolam ou congelam seletivamente parâmetros já consolidados tornam-se fundamentais para preservar o desempenho previamente alcançado, motivando abordagens construtivas com controle explícito de interferência entre estágios de aprendizado.

Uma ideia semelhante foi retomada por Smith *et al.* (2015), que introduziram o *gradual dropIn*, uma técnica que combina o aprendizado progressivo com mecanismos de regularização inspirados no *dropout*. Nesse método, novas camadas são incorporadas gradualmente por meio de uma interpolação entre representações rasas e profundas, permitindo que a rede adapte-se progressivamente ao aumento da profundidade. Essa inserção gradual atenua dificuldades relacionadas ao colapso do gradiente e promove convergência mais estável em redes muito profundas. Porém, as camadas previamente treinadas continuam sendo ajustadas, o que pode introduzir interferências retroativas e compromoter representações consolidadas.

Posteriormente, Miconi (2016) propôs um avanço conceitual ao introduzir o paradigma de *differentiable structure*, no qual tanto os pesos quanto a própria topologia da rede são otimizados via gradiente. Essa formulação permite que a arquitetura evolua de forma contínua, sem a necessidade de heurísticas de busca estrutural. Tal perspectiva aproxima o aprendizado de arquitetura do treinamento tradicional de parâmetros, representando um passo importante em direção a métodos mais automatizados de projeto de redes neurais. Embora a estrutura diferenciável permita a adaptação contínua da arquitetura, o ajuste estrutural muda o caminho do fluxo do gradiente, o que não garante comportamento estável do erro, ocasionando oscilações durante a evolução estrutural da rede.

No contexto de redes convolutivas, Badrinarayanan *et al.* (2017) apresentaram a *SegNet*, uma arquitetura de codificação e decodificação otimizada para segmentação semântica. Sua principal contribuição reside no uso dos índices de *max-pooling* do codificador durante a etapa de *upsampling* no decodificador, permitindo a recuperação eficiente da informação espacial sem a necessidade de armazenar mapas completos de ativação, o que reduz o custo computacional sem comprometer a acurácia. Embora não trate diretamente do crescimento arquitetural, a SegNet exemplifica como modularidade e reuso estrutural podem aprimorar o aprendizado profundo.

Por outro lado, Sun *et al.* (2017) introduziram o *generalized extreme learning machine autoencoder (GELMAE)*, combinando a rapidez de treinamento das máquinas de aprendizado extremo com a expressividade dos autoencoders. Essa fusão permitiu acelerar significamente a fase de aprendizado não supervisionado, apresentando resultados experimentais competitivos e, em alguns cenários, superiores a métodos tradicionais de agrupamento como *k-means* e *spectral clustering*. Nessa abordagem, ocorre maior comprometimento com a eficiência computacional ao custo de menor controle sobre a evolução estrutural.

O tema do crescimento adaptativo foi formalizado em Aghili e Mula (2020), onde

o aumento da profundidade da rede é modelado como um problema de controle ótimo. Nessa formulação, durante o treinamento, a estrutura é ajustada dinamicamente conforme a complexidade da tarefa, evitando tanto o subajuste quanto o sobreajuste. Essa visão reforça a ideia de que a profundidade ideal de uma rede não é fixa, mas dependente da distribuição dos dados e da fase de aprendizado.

Seguindo essa perspectiva adaptativa, Ashfahani *et al.* (2020) apresentaram o modelo *deep evolving denoising autoencoder* (DEVDAN), capaz de expandir e contrair sua estrutura de forma autônoma durante o treinamento. Diferentemente de abordagens estáticas, o DEVDAN adiciona ou remove unidades ocultas, explorando o crescimento em largura e não em profundidade, conforme a complexidade dos dados evolui, buscando equilibrar desempenho e custo computacional. Essa característica reforça a potencialidade do modelo entre as arquiteturas de aprendizado dinâmico.

Em um estudo complementar, Caillon e Cerisara (2021) demonstraram empiricamente que redes neurais que crescem progressivamente durante o treinamento tendem a alcançar mínimos mais planos na superfície de perda. Essa característica está associada a melhor capacidade de generalização e maior estabilidade de aprendizado, reforçando a importância do crescimento gradual em oposição à expansão abrupta da rede.

Na área de aprendizado contínuo, Kozal *et al.* (2022) propuseram mecanismos para aumentar a profundidade de forma dinâmica, com a criaçaõ de modelo em formato de árvore, garantindo que o conhecimento previamente adquirido seja preservado. Essa abordagem é particularmente relevante para cenários em que o modelo deve aprender sequencialmente, sem esquecer tarefas anteriores, um problema conhecido como *catastrophic forgetting*, e que o trabalho mitiga via isolamento estrutural/regularização entre blocos antigos e novos. Embora, essa abordagem apresente avanços significativos no que diz respeito à preservação do conhecimento, ainda mantêm o ajuste simultâneo de parâmetros previamente consolidados. Essa característica implica a presença de interferência retroativa residual, podendo gerar oscilações no erro e degradação pontual de representações aprendidas em estágios anteriores. Com isso, permanece em aberto a necessidade de estratégias que conciliem crescimento arquitetural progressivo com isolamento efetivo do conhecimento adquirido, garantindo estabilidade do erro e evitando a reotimização desnecessária de componentes já maduros.

De forma semelhante, Evci *et al.* (2022) desenvolveram o *GradMax*, um método que utiliza informações estatísticas de gradiente para decidir quando e onde expandir a rede. Essa técnica permite inserções estruturais controladas e adaptativas, orientadas por métricas internas de aprendizado, e não por regras fixas. Porém, torna-se vulnerável à localidade do grandiente e, como consequência, apresentar oscilações no erro durante o treinamento.

Além disso, Pan (2023) mostrou que o uso de inicialização ortogonal em redes em expansão melhora a estabilidade numérica e acelera a convergência, enquanto Zhu *et al.* (2023) exploraram a expansão dinâmica de camadas convolucionais durante o treinamento, permitindo

que a rede sofra adaptações a diferentes padrões de entrada sem a reinicialização completa do modelo, permanecendo com ajuste conjunto dos parâmetros existentes.

Mais recentemente, Wang *et al.* (2024) introduziram o *dynamic layer attention*, um mecanismo que permite interação dinâmica entre camadas por meio de blocos recorrentes, promovendo uma representação contextual mais rica, especialmente em tarefas de visão. Complementarmente, Hay e Ruder (2024) propuseram o *dynamic layer tying* para *transformers*, técnica que explora o compartilhamento dinâmico de pesos entre camadas recém-adicionadas, reduzindo a redundância paramétrica sem sacrificar desempenho.

Do ponto de vista teórico, Agarwal e Athalye (2024) reinterpretaram o processo de empilhamento de camadas como uma forma de gradiente acelerado, estabelecendo uma base matemática para entender o crescimento progressivo de redes como um processo de otimização contínua. Já Yang *et al.* (2025) introduziram o *layer expansion with stable activation* (LESA), uma estratégia para inicialização de novas camadas em modelos de linguagem de grande escala, garantindo estabilidade e preservação de desempenho durante a expansão arquitetural. Em complemento, Cao *et al.* (2025) aplicaram o transporte ótimo para inserir novas camadas preservando a função da rede-base, evitando degradação do conhecimento aprendido.

Radhakrishnan *et al.* (2025) propuseram o crescimento dinâmico de redes neurais do tipo MLP via gradiente descendente, adicionando neurônios durante o treinamento conforme a complexidade da tarefa. Essa abordagem melhora a generalização e a estabilidade, além de tornar o treinamento mais eficiente em comparação com redes estáticas. O método evidencia os benefícios do crescimento arquitetural incremental, alinhando-se ao objetivo do presente trabalho de inserção colaborativa de ramificações sem descaracterizar a rede original. Entretanto, essa abordagem realiza um crescimento modular dinâmico, em que uma nova camada é inserida com poucos neurônios e, após essa inserção, novas unidades são adicionadas a essa nova camada. Além disso, no ajuste dos pesos conectados aos novos neurônios, ocorrem também ajustes nos pesos das camadas anteriores, o que produz oscilações no aprendizado, embora essas oscilações sejam atenuadas pelo uso de uma máscara.

Além disso, estudos recentes reforçam a importância do crescimento adaptativo em SAE e redes profundas. Alfayez *et al.* (2023) propuseram o *dynamic depth learning* em SAE, no qual camadas são adicionadas progressivamente com base em métricas de desempenho e estabilidade do gradiente. A abordagem mostra ganhos significativos em generalização e robustez, demonstrando o potencial de redes que ajustam dinamicamente sua profundidade sem intervenção manual.

Berahmand *et al.* (2024) apresentaram uma revisão abrangente sobre autoencoders e suas aplicações em aprendizado de máquina, destacando que, embora tenham sido amplamente utilizados para redução de dimensionalidade e detecção de anomalias, a definição da topologia ideal ainda depende de heurísticas manuais, reforçando a necessidade de métodos construtivos e adaptativos.

Complementando esse panorama, Pedraza *et al.* (2024) exploraram o uso de autoencoders combinados com conceitos da teoria do caos para melhorar a detecção de exemplos adversariais. O estudo evidencia que a inserção incremental de camadas pode aumentar a robustez e a estabilidade da rede frente a perturbações externas, reforçando os benefícios do crescimento arquitetural progressivo.

Por fim, Fehring e Schmidhuber (2025) aplicaram os princípios de crescimento progressivo no contexto de aprendizado por reforço profundo, ajustando dinamicamente a estrutura da rede conforme a experiência acumulada. Essa integração entre crescimento arquitetural e aprendizado contínuo reforça o potencial do paradigma adaptativo em ambientes dinâmicos e não estacionários.

Sob a ótica dos trabalhos analisados, observa-se que as metodologias recentes convergem para o mesmo objetivo: aprimorar a capacidade de generalização, estabilidade e eficiência das redes neurais por meio de crescimento controlado e adaptação estrutural. Entretanto, nota-se que essas abordagens concentram-se na modificação direta da topologia interna da rede, isto é, novas camadas substituem ou acoplam-se estruturalmente às camadas anteriores. Até o momento, não foram identificadas estratégias que explorem a complementação colaborativa na saída da rede, em que múltiplos modelos coexistem e contribuem conjuntamente para o resultado final. Diante disso, o presente trabalho propõe uma estratégia de inserção colaborativa, em que ramificações são adicionadas sem afetar o resultado da estrutura original, promovendo aprendizado incremental, sinergia entre modelos e estabilidade de convergência. A metodologia correspondente é detalhada nos capítulos seguintes.

1.4 Contribuições do Trabalho

Este trabalho tem como principal contribuição a proposição de um algoritmo construtivo para redes neurais totalmente conectadas, integrado ao próprio processo de treinamento. Nessa abordagem, a arquitetura é expandida progressivamente por meio da inserção dinâmica de novas camadas via ramos durante a otimização, permitindo que a estrutura da rede evolua de acordo com a dinâmica da função de custo. O método estabelece uma separação explícita entre os parâmetros previamente consolidados e aqueles recém-inseridos, possibilitando a preservação do conhecimento já aprendido e reduzindo efeitos de interferência retroativa. Deste modo, as principais contribuições deste trabalho são listadas a seguir:

1. Proposição de um algoritmo construtivo para redes totalmente conectadas, com inserção incremental de camadas baseada em critérios objetivos de melhoria de desempenho;

2. Desenvolvimento de um mecanismo colaborativo de inserção de ramos (*branch learning*), permitindo o treinamento isolado de novos blocos enquanto os pesos previamente aprendidos permanecem preservados;

3. Definição de uma estratégia de inicialização seletiva dos novos pesos, reduzindo instabilidade no processo de otimização e mitigando efeitos de esquecimento catastrófico durante o crescimento da arquitetura;

4. Formalização dos modos de expansão estrutural, acompanhada de uma análise comparativa quanto à complexidade computacional e ao comportamento de convergência;

5. Implementação de um *software* com interface amigável para execução dos algoritmos desenvolvidos, visando facilitar treinamento e validação em diferentes contextos de dados e de hiperparâmetros de treinamento, com possibilidade de extensão por qualquer usuário.

1.5 Organização do trabalho

Este trabalho está organizado da seguinte forma. Na sequência deste capítulo, o Capítulo 2 apresenta a fundamentação teórica necessária para a compreensão da pesquisa, abordando redes neurais artificiais, autoencoders, stacked autoencoders e técnicas de treinamento, bem como métodos construtivos para redes neurais profundas. O Capítulo 3 descreve o método colaborativo proposto, detalhando sua formulação e o processo de inserção gradual de camadas. O Capítulo 4 apresenta os experimentos realizados, incluindo a descrição das bases de dados, o protocolo experimental, as métricas utilizadas e a análise dos resultados obtidos. Por fim, o Capítulo 5 discute as conclusões do trabalho, destacando as principais contribuições e perspectivas para trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os fundamentos teóricos que nortearam o desenvolvimento deste trabalho. Inicialmente, são discutidos os principais conceitos relacionados às redes neurais do tipo autoencoder, incluindo sua estrutura básica composta por codificador e decodificador, a função de perda associada à reconstrução e o papel do espaço latente na extração de representações internas. Também são abordadas variações relevantes, estratégias de treinamento, regularização e aspectos relacionados à capacidade de generalização dessas arquiteturas.

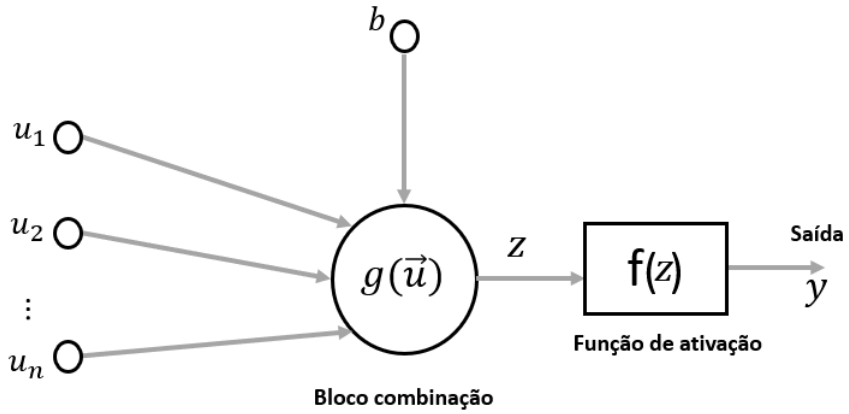
Na sequência, são apresentados os fundamentos das redes neurais construtivas, com ênfase nos algoritmos que realizam crescimento incremental da arquitetura ao longo do treinamento. São discutidos os princípios de adaptação topológica, critérios de inserção de novas unidades ou camadas, mecanismos de controle de complexidade e estratégias de atualização paramétrica durante a expansão estrutural. Além disso, são analisadas as implicações computacionais do crescimento progressivo da rede e sua relação com estabilidade, convergência e capacidade de representação.

2.1 Redes Neurais Artificiais

Uma rede neural artificial (RNA) é uma técnica computacional que possui um modelo matemático com inspiração nas estruturas neurais dos organismos. Como é sabido, o cérebro é capaz de realizar diversas atividades simultaneamente, podendo ser equiparado a um computador extremamente complexo por ser capaz de trabalhar de forma não-linear e em paralelo. As RNA possuem algumas semelhanças ao cérebro por serem capazes de adquirir conhecimento a partir do ambiente em que estão inseridas por meio de um processo de aprendizagem. Esse conhecimento é armazenado nos pesos sinápticos, que são localizados entre os nós da rede. Esses nós são conhecidos como neurônios artificiais, unidades processadoras ou simpesmente neurônios. As redes neurais possuem capacidade de realizar generalização, ou seja, podem produzir saídas adequadas com base em entradas que não estavam presentes em seu processo de treinamento (HAYKIN, 2001).

O modelo de funcionamento do neurônio artificial é ilustrado na Figura 1. Nessa representação, os sinais de entrada são denotados por $u_1, u_2, \ldots, u_n$, os quais são aplicados ao bloco de combinação. No bloco de combinação, processado por $g(\vec{u})$, os sinais de entrada são agregados em um valor escalar, ao qual pode ser somado um termo de polarização (*bias*) b, que atua como um deslocamento ajustável do modelo. A saída desse bloco é representada pela variável intermediária z, que representa a ativação linear do neurônio. Em seguida, o valor z é aplicado à função de ativação $f(z)$, responsável por introduzir não linearidade e por gerar a saída final do neurônio artificial, denotada por y.

Figura 1 – Estrutura de um neurônio artificial.



Fonte: Elaborada pelo autor.

2.1.1 Funções de Ativação

Conforme foi descrito, após o processo de combinação das entradas, é preciso passar o resultado pela função de ativação. Existem diversos tipos de funções de ativação de acordo com o objetivo da rede, inclusive também é possível usar diferentes funções de ativação em diferentes camadas. De acordo com Buduma e Lacascio (2017), além da função linear, as mais conhecidas são: sigmoide, tangente hiperbólica (tanh) e ReLU.

A função sigmoide comprime a entrada para o intervalo $(0, 1)$, sendo tradicionalmente utilizada nos neurônios das camadas escondidas. Quando usada na camada de saída, ela pode ser interpretada como uma probabilidade, desde que respeitadas suas leis e regras. Apesar de sua suavidade e sua derivabilidade, essa função apresenta limitações no processo de treinamento, sobretudo quando aplicada em camadas ocultas. Para valores elevados positivos ou negativos da entrada, sua resposta tende à saturação, reduzindo a eficiência do ajuste dos parâmetros. Além disso, por não ser centrada em zero, a sigmoide pode causar oscilações indesejadas no processo de otimização, resultando em um aprendizado mais lento (GOODFELLOW *et al.*, 2016).

A função tangente hiperbólica (*tanh*) pode ser vista como uma alternativa à sigmoide, pois mantém a não linearidade suave e produz saídas no intervalo $(-1, 1)$. Por ser centrada em zero, apresenta vantagens no processo de treinamento. No entanto, assim como outras funções suaves, a *tanh* apresenta regiões de saturação para valores elevados da entrada, o que motivou o desenvolvimento de funções de ativação alternativas, como a ReLU, mais adequadas para redes neurais profundas (GOODFELLOW *et al.*, 2016).

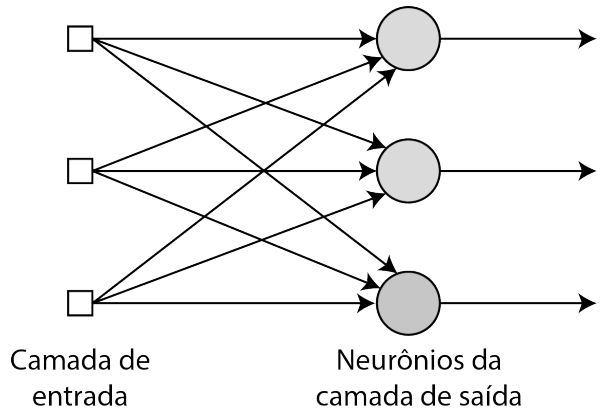
A função ReLU (*Rectified Linear Unit*) é uma função de ativação definida como nula para entradas negativas e linear para entradas não negativas. Sua forma simples resulta em baixo custo computacional e em um comportamento não saturante na região positiva. No entanto, para entradas negativas, a saída permanece constante em zero, o que pode levar

algumas unidades a tornarem-se permanentemente inativas durante o treinamento. Ainda assim, a ReLU é amplamente adotada como função de ativação em camadas ocultas de redes neurais profundas contemporâneas.

2.1.2 Arquiteturas das Redes Neurais Artificiais

Segundo Braga *et al.* (2000), as arquiteturas das RNA podem ser categorizadas de acordo com o número de camadas que a rede possui, o número de neurônios em cada uma das camadas, o tipo de conexão entre os neurônios e a topologia aplicada na rede. Em relação ao número de camadas, ela pode ser dividida entre camada única ou múltiplas, no primeiro tipo há o conjunto de sinais de entrada e estes estão conectados diretamente aos neurônios de saída, ou seja, a camada de entrada está diretamente conectada apenas à camada de saída. Como os nós de entrada não são contados como neurônios, logo existem apenas os neurônios de saída. Devido a isto, ela é chamada de camada única, como ilustrado na Figura 2.

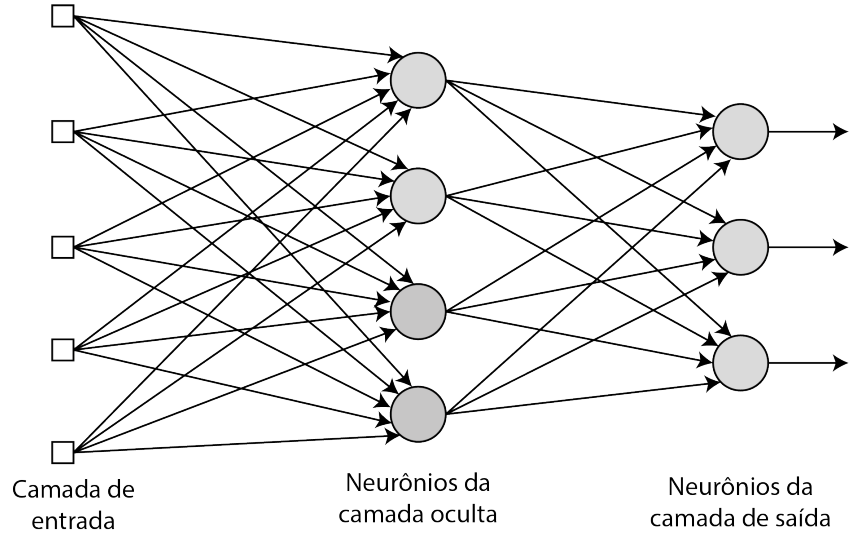
Figura 2 – Arquitetura de camada única de uma rede neural artificial.



Fonte: Elaborada pelo autor.

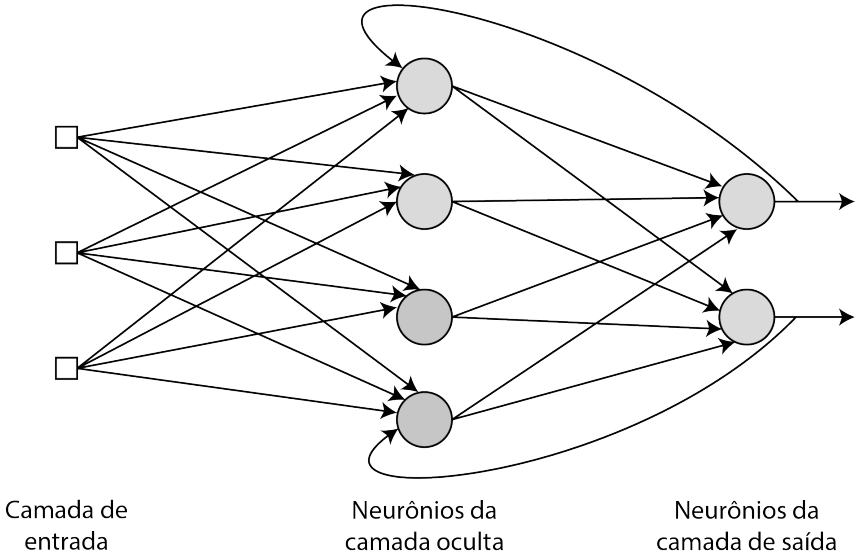
Existem as chamadas de camadas ocultas nas redes de múltiplas camadas, elas servem para aumentar a quantidade de conexões sinápticas e podem aumentar a capacidade de processamento dos dados recebidos, este tipo de arquitetura torna-se mais interessante quando o tamanho da camada de entrada é grande. A quantidade de camadas ocultas que é possível colocar na rede é variável e depende de quem está projetando ela. Usualmente, a camada de entrada é conectada à primeira camada oculta, que conecta-se à seguinte até que chegue-se na camada dos neurônios de saída, isto pode ser observado na Figura 3. Conforme foi citado, a rede também pode ser classificada de acordo com o tipo de conexões existentes, ela é conhecida como totalmente conectada quando cada um dos neurônios de uma camada está conectado com os neurônios da camada seguinte, contudo quando nem todos estão conectados, a rede é chamada de parcialmente conectada (HAYKIN, 2001).

Figura 3 – Arquitetura de multicamada de uma rede neural artificial.



Fonte: Elaborada pelo autor.

Figura 4 – Arquitetura de uma rede neural artificial com realimentação.



Fonte: Elaborada pelo autor.

As arquiteturas e as figuras apresentadas seguiram um tipo de conexão chamado *feedfoward* ou acíclica, isto significa que cada camada está conectada com a camada seguinte e as saídas destas não estão conectadas com camadas anteriores, ou seja, as informações seguem apenas um sentido. Todavia, também existem as conexões do tipo *feedback* ou cíclicas, cuja ligação de retorno é mais conhecida como realimentação. Quando uma rede possui realimentação, significa que a saída de uma camada pode servir como entrada para uma camada anterior à sua, isto ocorre nas chamadas de redes neurais recorrentes, em que um neurônio de uma camada alimenta o neurônio de uma camada anterior. Um exemplo de arquitetura de uma rede neural com uma realimentação pode ser observado na Figura 4, onde

existem ligações da saída da rede para a entrada da camada oculta (BRAGA *et al.*, 2000).

2.1.3 Aprendizado da Rede

De acordo com Haykin (2001), ocorre um processo interativo entre a rede e o ambiente, de forma que a rede fica mais instruída acerca deste e faz os ajustes dos pesos sinápticos. Essa mudança nos parâmetros da rede para adaptar-se ao ambiente, onde ela está inserida, é chamado de processo de aprendizagem. Esse processo pode ser dividido em três partes: na primeira a rede é estimulada pelo ambiente, em seguida ela sofre modificações em seus parâmetros livres devido às estimulações e, por fim, responde de uma nova forma ao ambiente, de acordo com as alterações dos parâmetros.

Existem tipos de aprendizados das redes neurais. Os mais comuns são chamados aprendizado supervisionado, não-supervisionado e semi-supervisionado. No primeiro tipo, utiliza-se um elemento externo para avaliar o grau de proximidade da saída produzida pela rede em relação à solução ideal. Desta forma, a rede pode realizar alterações nos pesos sinápticos para diminuir a taxa de erro. Já no aprendizado não-supervisionado, a rede não tem conhecimento dos dados de saída corretos. Seu processo de funcionamento consiste em fazer agrupamentos ou reconhecer a vizinhança dos dados, desta forma, os ajustes são efetuados de acordo com as características estatísticas dos dados apresentados para otimizar os parâmetros livres das redes. O aprendizado semi-supervisionado é usado para implementar o treinamento de classificadores quando uma pequena parte da base de dados possui os rótulos de classificação, enquanto que o restante não possui, desta forma é possível realizar um melhor aproveitamento da base de dados (VIANA, 2019).

O aprendizado por reforço constitui outro paradigma amplamente conhecido. Nesse caso, a rede interage com o ambiente por meio da execução de ações, recebendo recompensas quando estas resultam em comportamentos desejáveis e penalidades quando produzem resultados indesejados. A partir desse mecanismo de realimentação, o modelo ajusta sua política de decisão de modo a maximizar a recompensa acumulada ao longo do processo de aprendizagem (GERON, 2017).

Usando como exemplo um processo de aprendizado do tipo supervisionado em uma rede neural, durante o treinamento, a rede irá realizar uma determinada quantidade de exemplos e ajustar seus pesos com o objetivo de minimizar os erros até que esteja com um valor aceitável. É possível perceber que o objetivo no treinamento é tentar tornar o erro o mais próximo de zero possível, contudo, dependendo da complexidade da rede em que se esteja trabalhando, pode ser mais complexo de ser atingido. Tomando os valores do erro e dos pesos como eixos de um gráfico, seria possível formar uma superfície utilizando todas as combinações possíveis entre os pesos e o erro, com o valor mínimo de erro sendo o menor ponto da superfície. Em cada iteração do treinamento, a rede tenta dar um pequeno passo para a direção deste ponto de erro mínimo, este processo é chamado de "descida do gradiente"e ele é um dos fatores aplicados

nos algoritmos para fazer os ajustes dos pesos sinápticos no treinamento da rede neural. Com o objetivo de melhorar a eficiência do treinamento da rede, o gradiente é multiplicado por um fator conhecido como "taxa de aprendizado", que deve ser definido com cuidado, pois caso seja muito pequeno, poderá aumentar o tempo de treinamento e, caso seja muito grande, poderá divergir do valor mínimo de erro (BUDUMA; LACASCIO, 2017).

Um algoritmo bastante conhecido que é usado para o treinamento supervisionado é o chamado algoritmo de retropropagação (do inglês, *backpropagation* - BP). Esse treinamento ocorre em duas fases chamadas de *forward* e *backward*. A primeira é utilizada para a rede produzir uma saída de acordo com os seus pesos atuais e os sinais de entrada. Na segunda fase, a saída encontrada pela rede é comparada com a saída desejada, resultando em um valor de erro. Esse erro, com base na descida do gradiente, é usado na atualização dos pesos sinápticos (BRAGA *et al.*, 2000). Com o objetivo de melhorar o processo de convergência da rede, foram realizadas algumas variações no algoritmo *backpropagation*, tais como o método de inserção do termo de *momentum*, o método *resilient-propagation* e o *Levenberg-Marquardt* (SILVA *et al.*, 2016).

No entanto, a forma clássica do algoritmo de retropropagação (*backpropagation* – BP) apresentada anteriormente utiliza a descida do gradiente por lote (do inglês, *batch gradient descent*), na qual a atualização dos pesos é realizada apenas após a avaliação de todo o conjunto de amostras de treinamento. Essa estratégia torna o processo de treinamento computacionalmente mais lento e pode levar à estagnação em regiões da superfície de erro associadas a pontos de sela, caracterizados por gradiente nulo, mas que não correspondem a mínimos locais.

Como alternativa, foi proposta a descida do gradiente estocástica (do inglês, *stochastic gradient descent* – SGD), na qual a atualização dos pesos ocorre a cada amostra de treinamento. Essa abordagem introduz ruído no processo de otimização, tornando a dinâmica de treinamento mais eficiente e reduzindo a probabilidade de permanência em mínimos locais.

Uma solução intermediária amplamente adotada é a descida do gradiente por mini lotes (do inglês, *minibatch gradient descent*), em que a atualização dos pesos é realizada após a apresentação de um subconjunto de amostras. O tamanho do mini lote é definido como um hiperparâmetro do modelo, permitindo um compromisso entre estabilidade numérica e eficiência computacional (BUDUMA; LACASCIO, 2017).

Para aperfeiçoar o treinamento das RNA, é possível realizar a adaptação da taxa de aprendizado através dos chamados "otimizadores". Um deles é o AdaGrad (do inglês, *Adaptive Gradient Algorithm*), que utiliza uma acumulação do histórico de gradientes para adaptar a taxa de aprendizado. Entretanto, esse algoritmo possui a tendência de realizar uma diminuição prematura na taxa, o que, em situações práticas, pode ser ineficiente em problemas mais complexos. Outro otimizador é o RMSProp (do inglês, *Root Mean Square Prop*), que utiliza um fator de decaimento para limitar a quantidade do histórico de gradientes que seriam mantidos,

fazendo com que o RMSProp apresente mais eficiência e seja bastante utilizado. Um otimizador bastante popular é o Adam (do inglês, *Adaptive Moment Estimation*), que utiliza dois fatores para ajustar o decaimento do gradiente, um para corrigir o mais recente e outro para o histórico. Em certos casos, o Adam consegue ser ainda mais efetivo do que o RMSProp (BUDUMA; LACASCIO, 2017).

Durante o processo de treinamento, em certos casos, pode ocorrer que o modelo da rede neural desenvolvido não seja capaz de generalizar, ou seja, o modelo consegue aprender apenas os casos que foram utilizados como exemplo e não é capaz de replicar para outros diferentes dos observados no treinamento. Na literatura, esse caso é chamado de *overfitting* e acaba sendo um desafio no campo do aprendizado de máquina. Para verificar se esse problema está ocorrendo é separado a base de dados em treinamento, validação e teste, desta forma é possível verificar se o modelo empregado é capaz de generalizar para os casos não vistos durante o treinamento. Adicionalmente, o acompanhamento do desempenho ao longo das épocas de treinamento possibilita identificar o início do *overfitting*, viabilizando a interrupção antecipada do treinamento por meio da técnica conhecida como *early stopping*. (BUDUMA; LACASCIO, 2017).

Outra estratégia amplamente empregada para mitigar o *overfitting* é o uso do *dropout*. Essa técnica consiste em desativar aleatoriamente unidades da rede durante o processo de treinamento, segundo uma determinada probabilidade, a cada época. Com isso, reduz-se a dependência excessiva de neurônios individuais ou de pequenos subconjuntos, incentivando o aprendizado de representações mais robustas distribuídas ao longo da rede. Como consequência, melhora-se a capacidade da rede de conseguir a generalização (SRIVASTAVA *et al.*, 2014).

2.2 *Stacked autoencoder*

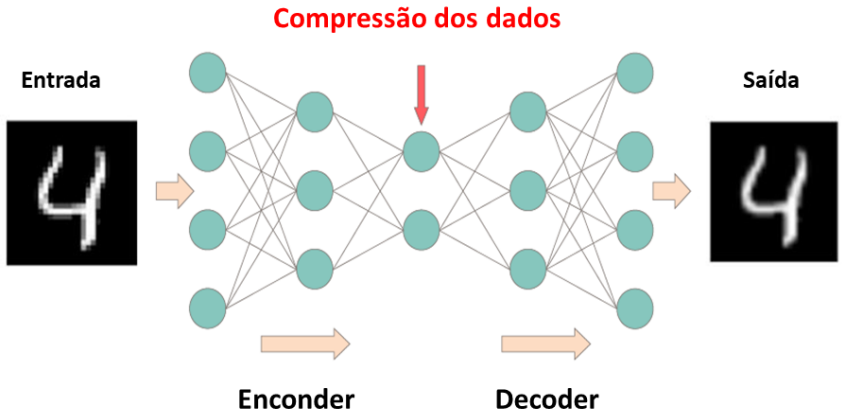
Os autoencoders são uma rede que usa uma técnica de aprendizado supervisionado, na qual as redes neurais são usadas para a tarefa de aprendizado de representação. Especificamente, é projetada uma arquitetura de rede neural de modo a impor um gargalo na rede capaz de forçar uma representação compactada da entrada original com o menor erro de reconstrução possível(CHEN *et al.*, 2017). Se os recursos de entrada fossem independentes um do outro, essa compressão e reconstrução subsequente seriam uma tarefa muito difícil. Entretanto, se houver algum tipo de estrutura nos dados (ou seja, correlações entre os recursos de entrada), essa estrutura poderá ser aprendida e consequentemente aproveitada ao forçar a entrada através do gargalo da rede (AURÉLIEN, 2017; DENG; YU, 2014).

Para a detecção de novidades por esta técnica, o escore de dissimilaridade mais comum é o erro quadrático de reconstrução do dado sob análise, visto que dados espúrios e de classes desconhecidas tendem a não ser adequadamente representados por tais mapeamentos, logo estariam associados a maiores erros (PIMENTEL *et al.*, 2014). Nesse contexto, conforme ilustra

a Figura 5, esse tipo de rede é composto de duas partes (SCACCIA, 2020; BANK *et al.*, 2020):

- Codificador (Encoder): bloco inicial da rede, responsável pela extração de correlações úteis dos atributos de entrada. Com isso, cria uma representação interna de menor dimensionalidade desses dados de entrada e com informação suficiente para a posterior restauração dos dados (realiza a extração de atributos).
- Decodificador (Decoder): bloco final da rede, toma como entrada a representação interna criada pelo encoder e a partir dela tenta reconstruir os dados originais de entrada com o mínimo de perda possível (parte regenerativa/reconstrutora).

Figura 5 – Modelo básico de uma autoencoder.



Fonte: Elaborado pelo autor (2026).

2.2.1 Características

Se o único objetivo dos autoencoders fosse copiar a entrada para a saída, eles seriam inúteis. De fato, espera-se que, treinando o autoencoder para copiar a entrada para a saída, a representação latente tenha propriedades úteis. Isso pode ser conseguido criando restrições na tarefa de cópia (ZHUANG *et al.*, 2015; SCACCIA, 2020).

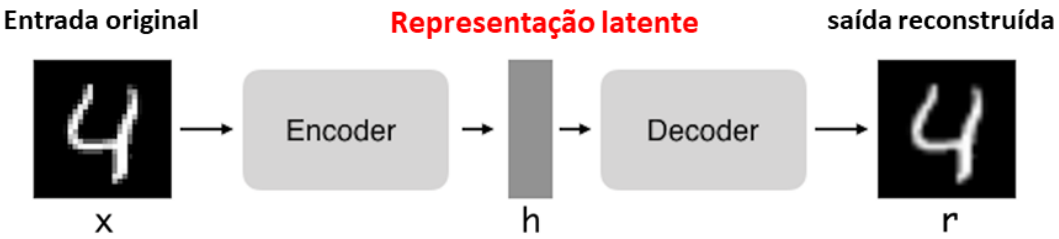
Uma forma de extrair representações relevantes por meio de autoencoders consiste em restringir a camada oculta **h**, ilustrada na Figura 6, a possuir dimensão inferior à da entrada **x**. Nesse caso, o autoencoder é denominado incompleto. Ao impor essa restrição, o modelo é forçado a capturar apenas os aspectos mais relevantes dos dados de treinamento, evitando soluções triviais de cópia direta da entrada para a saída.

Por outro lado, quando o autoencoder dispõe de capacidade excessiva, o aprendizado pode limitar-se à simples reconstrução da entrada, sem a extração de informações úteis sobre a distribuição dos dados. Esse comportamento pode ocorrer quando a dimensão da representação latente é igual ou superior à dimensão da entrada, incluindo o caso de autoencoders com excesso

de completude. Nessas situações, mesmo arquiteturas com codificadores e decodificadores lineares são capazes de reproduzir a entrada na saída sem aprender representações significativas.

De modo geral, a eficácia de um autoencoder depende do equilíbrio entre a dimensão do código latente e a capacidade das redes codificadora e decodificadora, que devem ser escolhidas de acordo com a complexidade da distribuição dos dados a ser modelada.

Figura 6 – Características de uma rede autoencoder.



Fonte: Adaptado de Hubens (2018).

2.2.2 Uso dos autoencoders

Atualmente, o *denoising* de dados (remoção de ruídos) e a redução de dimensionalidade para visualização de dados são considerados duas principais aplicações práticas interessantes de autoencoders, possuindo aprendizado de projeções de dados mais interessantes que o PCA ou outras técnicas básicas (AURÉLIEN, 2017; SCACCIA, 2020; GONDARA, 2016).

A partir dos exemplos de dados, ocorre o aprendizado automático dos autoencoders. Com isso, torna-se fácil treinar instâncias especializadas do algoritmo que oferecem bom desempenho em um tipo específico de entrada e que não necessitem e nenhuma nova engenharia de recursos, apenas os dados de treinamento apropriados.

Outra aplicação dos autoencoders é como tarefa preliminar ao reconhecimento de imagens com Redes Neurais Convolucionais (CNN). Observando a Figura 6, tem-se que a saída do autoencoder é a imagem do número quatro de modo mais suave. Ou seja, aplica-se os autoencoders para remover ruído dos dados (*denoising*), sendo a saída dos autoencoders utilizada para o treinamento de um um modelo de CNN (ULLAH *et al.*, 2019; RUSSO *et al.*, 2020).

Os autoencoders são treinados para preservar o máximo de informações possível quando uma entrada é passada pelo codificador e depois pelo decodificador, mas também são treinados para produzir novas representações com propriedades agradáveis. Nesse contexto, diferentes tipos de autoencoders visam atingir diferentes tipos de propriedades, conforme será tratado na próxima seção.

Cabe destacar que, após o treinamento, é comum utilizar apenas o codificador como extrator de características, descartando o decodificador, especialmente em tarefas

supervisionadas como classificação e regressão. Nesses cenários, o interesse principal está nas representações internas aprendidas pelo modelo, e não na capacidade de reconstrução do sinal original. Essa prática é amplamente adotada na literatura de aprendizado de representações, na qual os autoencoders são utilizados como mecanismos de redução de dimensionalidade e transformação de dados em espaços mais discriminantes (HINTON; SALAKHUTDINOV, 2006; BENGIO *et al.*, 2013; GOODFELLOW *et al.*, 2016).

Adicionalmente, a remoção do decodificador contribui para a redução da complexidade computacional do modelo, tornando-o mais eficiente tanto em termos de treinamento quanto de inferência. Mesmo sem a etapa explícita de reconstrução, o modelo mantém sua utilidade em diferentes aplicações, podendo ser integrado a arquiteturas maiores ou adaptado para diversas tarefas de aprendizado supervisionado e não supervisionado. Dessa forma, os autoencoders deixam de ser apenas modelos de reconstrução e passam a atuar como ferramentas versáteis dentro do contexto mais amplo de aprendizado profundo.

2.3 Principais Tipos de Redes Neurais Artificiais autoencoders

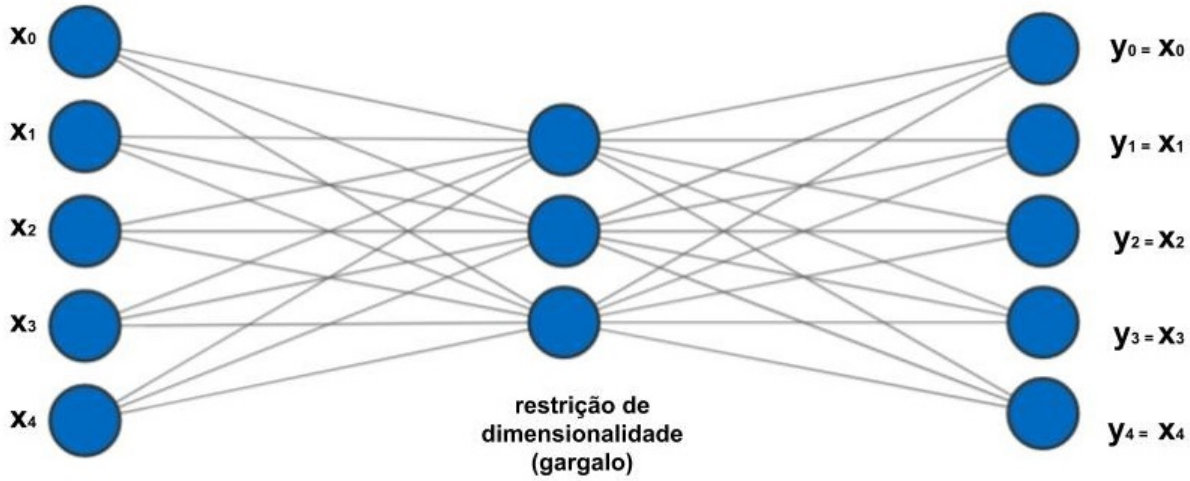
Os autoencoders codificam os valores de entrada **x** usando uma função **f**. Em seguida, decodificam os valores codificados **f(x)** usando uma função **g** para criar valores de saída idênticos aos valores de entrada. O objetivo do autoencoder é minimizar o erro durante o processo de reconstrução entre a entrada e a saída. Com isso, conseguem aprender os recursos importantes presentes nos dados. Quando uma representação permite uma boa reconstrução de sua entrada, ela retém grande parte das informações presentes na entrada (DOERSCH, 2016; KINGMA; WELLING, 2019).

2.3.1 Autoencoder padrão

Na sua forma mais simples, o autoencoder é uma rede neural artificial com três camadas, isto é, uma rede neural formada por uma camada de entrada, uma oculta e uma camada de saída, conforme ilustra a Figura 7. A entrada é aprendida a ser reconstruída na saída, por exemplo, usando o otimizador Adam, entre outras opções, e a função de perda de erro quadrático médio (DONG *et al.*, 2018; SCACCIA, 2020).

A idéia central é fazer com que a rede produza na camada de saída os mesmos valores que foram alimentados na camada de entrada, porém são acrescentados restrições nas camadas internas para evitar que a rede simplesmente faça uma replicação do valores da entrada, gerando eficientes aprendizados sobre representações internas. Nesse contexto, a restrição mais comum é limitar o número de unidades nas camadas ocultas intermediárias, conseguindo criar um tipo de gargalo no centro da rede.

Figura 7 – Modelo do autoencoder padrão.

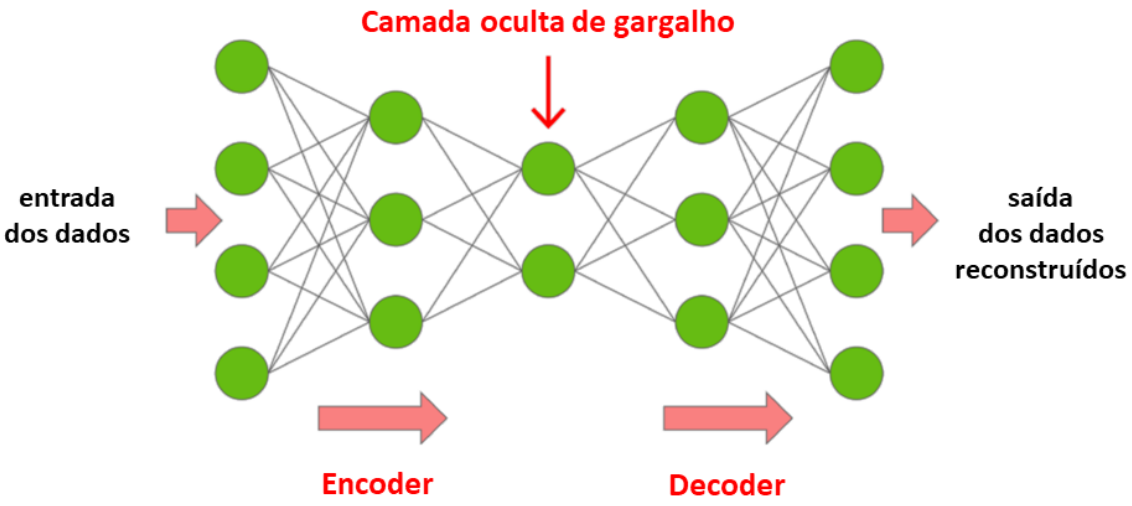


Fonte: (SCACCIA, 2020).

2.3.2 Autoencoder multicamada

Quando uma camada oculta não for suficiente, pode-se estender o autoencoder para mais camadas ocultas, conforme ilustra a Figura 8. Assim, por exemplo, uma possível implementação poderia usar 3 camadas ocultas em vez de apenas uma. Com isso, qualquer uma das camadas ocultas pode ser escolhida como representação de recurso, no entanto, o ideal é produzir uma rede simétrica e usar a camada mais intermediária.

Figura 8 – Modelo do autoencoder multicamada.



Fonte: Elaborada pelo autor.

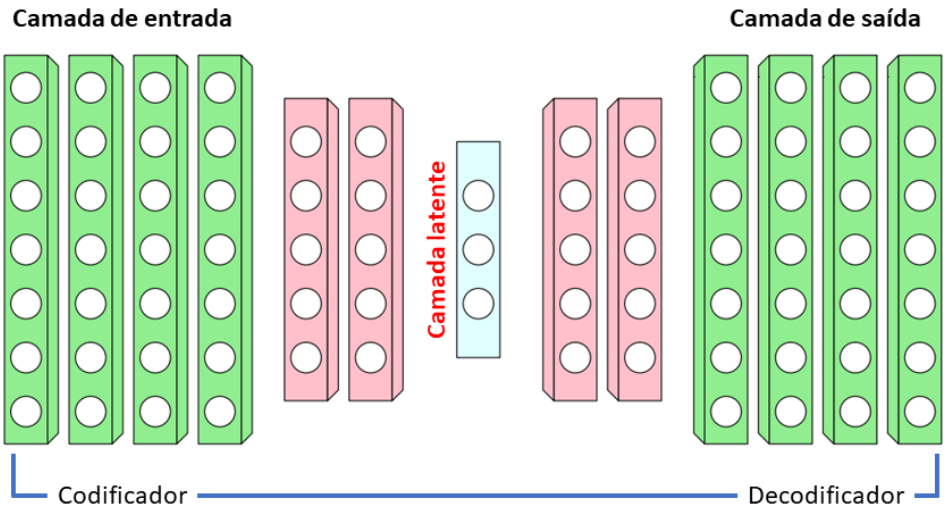
2.3.3 Autoencoder convolucional

Esta rede é formada por múltiplas camadas tipicamente simétricas. De modo simplista, possui tamanho (dimensão) reduzido progressivamente até a camada latente. De modo simétrico

à entrada, as dimensões são aumentadas até a camada de saída (BOUTARFASS; BESSERER, 2019). A Figura 9 ilustra uma autoencoder convolucional.

Os autoencoders podem ser usados com convoluções em vez de camadas totalmente conectadas. Para isso, usam-se imagens (vetores 3D) em vez de vetores 1D achatados (*flattened*). Então, diminui-se a imagem de entrada para fornecer uma representação latente com dimensões menores e forçar o autoencoder a aprender uma versão compactada das imagens.

Figura 9 – Modelo do autoencoder convolucional.



Fonte: Adaptado de Honorato *et al.* (2020).

2.3.4 Autoencoder regularizado

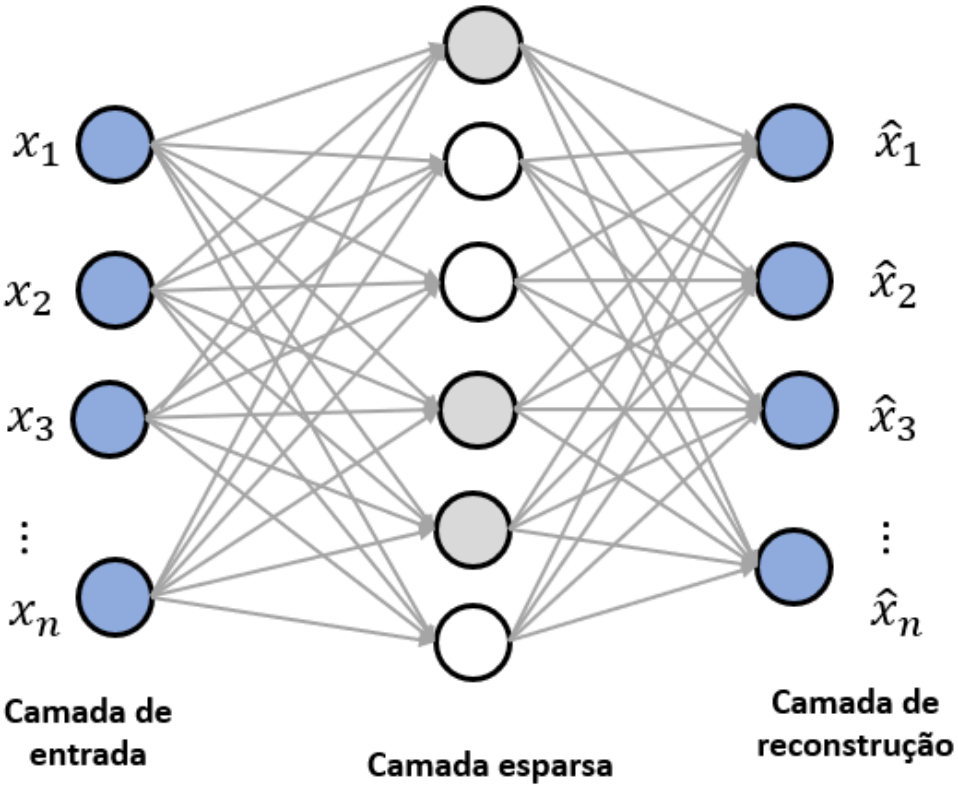
Existem outros modos pelos quais pode-se restringir a reconstrução de um autoencoder, além de impor uma camada oculta de menor dimensão que a entrada. Em vez de limitar a capacidade do modelo mantendo o codificador e o decodificador rasos e o tamanho do código pequeno, os autoencoders regularizados utilizam uma função de perda que fornece para o modelo propriedades adicionais além da capacidade de copiar sua entrada para sua saída. Na prática, geralmente são dois tipos de autoencoder regularizado: o autoencoder esparso e o autoencoder *denoising*.

2.3.5 Autoencoder esparso

Os autoencoders são geralmente treinados por meio do algoritmo de retropropagação do erro, ajustando os parâmetros do codificador e do decodificador de forma a minimizar o erro de reconstrução entre a entrada e a saída. A redução de dimensionalidade não é obtida pela retropropagação em si, mas pela imposição de restrições à representação latente ou à estrutura do modelo. Para isso, um pequeno subconjunto de neurônios permanece ativo para

cada padrão de entrada. Como ilustrado na Figura 10, essa abordagem favorece à aprendizagem de características mais discriminativas e interpretáveis.

Figura 10 – Autoencoder esparso.



Fonte: Elaborada pelo autor.

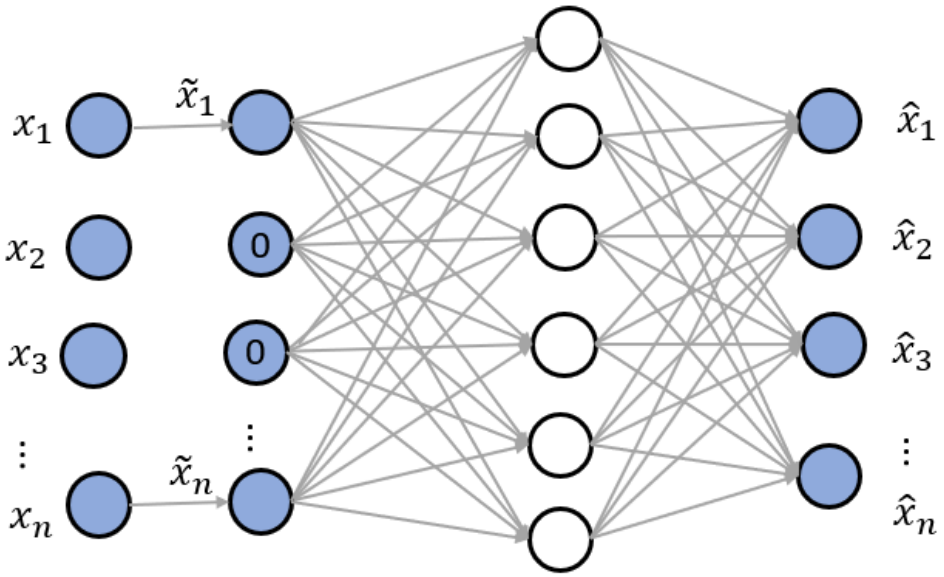
Autoencoders esparsos são comumente empregados como mecanismos de aprendizado de representações para tarefas subsequentes, como classificação. Ao impor uma restrição de esparsidade, o modelo é incentivado a responder a padrões estatísticos específicos do conjunto de dados, em vez de simplesmente aprender uma função de identidade. Dessa forma, embora o treinamento seja formulado como uma tarefa de reconstrução da entrada, a introdução de uma penalidade de esparsidade conduz à aprendizagem de características relevantes como um subproduto do processo de otimização.

Outra forma de limitar a reconstrução do autoencoder é impor uma restrição à sua perda, como exemplo, adicionando um termo de reguralização na função de perda que seja capaz de fazer com que ocorra o aprendizado da representação esparsa de dados. Na camada oculta, pode-se adicionar um regularizador de atividades L1, que aplicará uma penalidade na função de perda durante a fase de otimização. Como isso, a representação será mais esparsa.

2.3.6 Autoencoder denoising

O autoencoder *denoising* é uma variante do autoencoder tradicional que introduz ruído controlado nos dados de entrada durante o treinamento, com o objetivo de aprender representações mais robustas. Em vez de adicionar uma penalidade explícita à função de perda, essa abordagem modifica o próprio termo de erro de reconstrução, treinando o modelo para recuperar a versão original dos dados a partir de uma entrada corrompida. Dessa forma, o autoencoder é forçado a capturar estruturas estáveis e informativas da distribuição dos dados, evitando soluções triviais baseadas apenas na cópia da entrada (GONDARA, 2016). O processo de *denoising* pode ser implementado por meio de um mapeamento estocástico que adiciona ruído à entrada bruta antes de fornecê-la à rede. A Figura 11 ilustra a arquitetura de um autoencoder do tipo *denoising*.

Figura 11 – Autoencoder *denoising*.



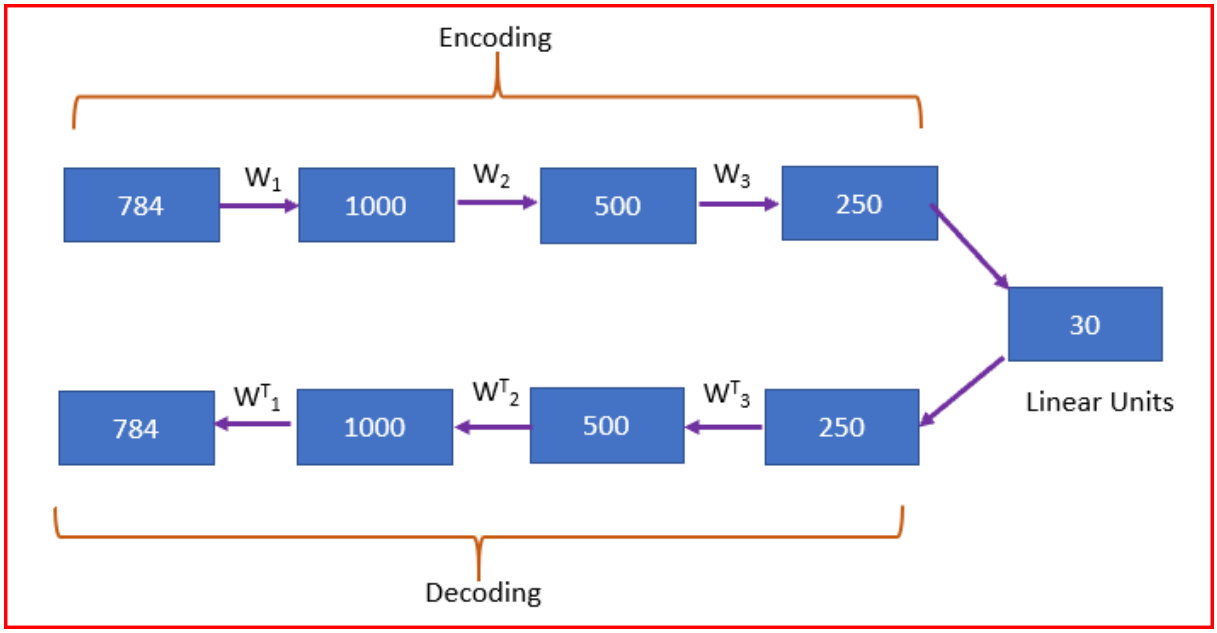
Fonte: Elaborada pelo autor.

2.3.7 *Contractive* autoencoders

O objetivo do autoencoder contrativo (CAE) é ter uma representação aprendida robusta, menos sensível a pequenas variações nos dados. A robustez da representação para os dados é obtida com a aplicação de um termo de penalidade à função de perda. O termo da penalidade é a norma Frobenius da matriz jacobiana para a camada oculta, calculada em relação à entrada. A norma de Frobenius da matriz jacobiana é a soma do quadrado de todos os elementos.

O CAE supera os resultados obtidos pela regularização do autoencoder usando decaimento de peso ou *denoising*. O CAE é boa escolha em relação ao autoencoder *denoising* para aprender a extrair recursos úteis. O termo de penalidade produz mapeamento que contrai fortemente os dados e, portanto, o nome autoencoder Contrativo.

Figura 12 – *Modelo do deep belief networks.*



Fonte: (ACADEMY, 2021).

2.3.8 *Deep Belief Networks*

As *deep belief networks* (DBN), ou redes de crença profunda, também conhecidas como máquinas de Boltzmann profundas, constituem modelos probabilísticos generativos formados pela composição de múltiplas Máquinas de Boltzmann Restritas (RBM). Cada RBM é estruturada em uma camada visível e uma camada oculta, cujos neurônios são conectados de forma bidirecional. Entretanto, devido essa estrutura restrita, não existem conexões entre neurônios pertencentes à mesma camada, o que simplifica o processo de treinamento.

O algoritmo de treinamento de uma DBN é composto por duas etapas distintas: um pré-treinamento não supervisionado e um ajuste fino supervisionado. Na primeira etapa, adota-se uma estratégia de treinamento *greedy* camada a camada, na qual cada RBM é treinada de forma independente. Esse procedimento resulta em inicializações baseadas em soluções locais sucessivas que, embora não garantam a obtenção de um ótimo global, são adequadas para inicializar redes profundas.

Na segunda etapa, a rede previamente pré-treinada é desenrolada e submetida ao ajuste fino supervisionado, no qual os parâmetros de todas as camadas são refinados de maneira conjunta por meio do algoritmo de retropropagação, utilizando dados rotulados e um critério de erro associado à tarefa final.

A Máquina de Boltzmann Restrita (RBM) constitui, portanto, o elemento fundamental das *Deep Belief Networks*. Conforme ilustrado na Figura 12, uma imagem de entrada com 784 pixels pode ser processada por uma pilha de RBMs treinadas sequencialmente, resultando, após o ajuste fino supervisionado, em uma representação codificada mais compacta e computacionalmente

eficiente.

2.3.9 Deep autoencoders

Deep autoencoders podem ser interpretados como arquiteturas profundas compostas por duas redes simétricas: uma responsável pela codificação e outra pela decodificação. Essas redes são frequentemente inicializadas por meio de estruturas análogas às *Deep Belief Networks*, empregadas no treinamento camada a camada. Tipicamente, autoencoders profundos apresentam entre quatro e cinco camadas no codificador, seguidas por um número equivalente de camadas no decodificador, formando uma arquitetura aproximadamente simétrica em torno da representação latente (SØNDERBY *et al.*, 2016; ZHUANG *et al.*, 2015; ZHANG *et al.*, 2018).

Deep autoencoders são criados através do empilhamento de autoencoders (Stack auto-encoders). O treinamento da arquitetura é realizado uma camada de cada vez. Quando finalizado esse processo, uma camada de classificação é adicionada e a rede profunda pode ser ajustada. A vantagem de usar mais camadas ocultas do que um único auto-codificador é que os dados de entrada de alta dimensão podem ter sua complexidade reduzida.

Deep autoencoders são construídos por meio do empilhamento progressivo de múltiplos autoencoders, formando arquiteturas conhecidas como *stacked autoencoders*. O pré-treinamento não supervisionado é realizado de maneira incremental, camada a camada, permitindo que cada novo bloco aprenda representações complementares às já obtidas, enquanto preserva o conhecimento previamente adquirido. Após a conclusão desse processo, a arquitetura completa pode ser ajustada de forma conjunta ou estendida com camadas adicionais voltadas à tarefa final (como exemplo, tarefa de classificação).

O uso de múltiplas camadas ocultas possibilita uma redução gradual da complexidade de dados de alta dimensão, promovendo a extração de representações hierárquicas cada vez mais abstratas. Essa estratégia de crescimento progressivo fornece a base conceitual para métodos construtivos, nos quais a profundidade da rede é adaptada de acordo com a complexidade dos dados e com critérios de desempenho previamente definidos.

2.4 Redes Neurais Construtivas

2.4.1 Abordagens construtivas incrementais

Em abordagens tradicionais (por exemplo, a MLP), o processo de treinamento da rede exige a definição prévia da arquitetura (número de camadas ocultas e de neurônios) antes do treinamento dos pesos sinápticos da rede. Com isso, invariavelmente a definição da arquitetura acaba sendo feita de forma arbitrária pelo projetista. Assim, fica aumentado o risco da arquitetura não ser a mais adequada para o problema, resultando em um aprendizado

deficiente. Já no caso dos algoritmos construtivos, a definição da arquitetura da rede é feita automaticamente durante o treinamento, o qual também realiza o ajuste dos pesos sinápticos. Há também uma tendência de que os algoritmos construtivos forneçam arquiteturas mais parcimoniosas em termos de número de neurônios e conexões sinápticas.

Entre os principais aspectos dos algoritmos construtivos, destacam-se os seguintes:

- Número de neurônios adicionados: refere-se à quantidade de neurônios que são adicionados na rede a cada iteração do processo construtivo. Na maioria dos casos, é feita a adição de um neurônio a cada iteração.
- Padrão de conexões do novo neurônio: refere-se a um tipo de política de conexão do novo neurônio a ser adicionado à rede. Por exemplo, o novo neurônio será conectado com todos os neurônios de todas as camadas ocultas prévias, ou apenas com os neurônios da camada de entrada e da camada imediatamente anterior.
- Inserção e/ou poda: os algoritmos construtivos englobam não só algoritmos que partem de uma arquitetura mínima e buscam incrementar componentes (neurônios e conexões) na rede, mas também etapas, que podem levar à redução do número de componentes da arquitetura (poda), caso esta redução implique uma melhora de desempenho da rede.
- Direção do crescimento da rede: destacam-se três casos. O primeiro tem crescimento a partir da camada de entrada até a camada de saída. No segundo caso, a partir da camada de saída até a camada de entrada. Por fim, todas aquelas estratégias que não encaixam-se nas duas primeiras compõem o terceiro caso.
- Funcionalidade dos neurônios inseridos: embora a maioria dos algoritmos construtivos não faça diferença entre os neurônios ocultos, existem alguns que consideram a criação de camadas nas quais um neurônio assume um papel de maestro e os demais de auxiliares.
- Ajuste dos pesos sinápticos do novo neurônio: a maioria dos algoritmos construtivos adota técnicas de ajuste de pesos locais e que só ajustam as novas conexões que o novo neurônio inserido na rede traz consigo
- Critério de parada: o critério de parada mais utilizado é o desempenho da rede, seja em termos de taxa de acertos/erro para problemas de classificação ou a minimização quantitativa do erro, como o erro quadrático médio para problemas de regressão de dados. O processo construtivo é concluído quando estes valores permanecem constantes ou com variação não-significativa entre duas ou mais iterações. Em casos especiais de restrições de uso de memória, o critério de parada pode ser complementado com a definição de um número máximo de neurônios a ser atingido pelo algoritmo construtivo
- Forma da rede: refere-se à configuração topológica que a rede vai adquirindo à medida que vai sendo construída. Por exemplo, existem formas de pirâmide e torre com apenas

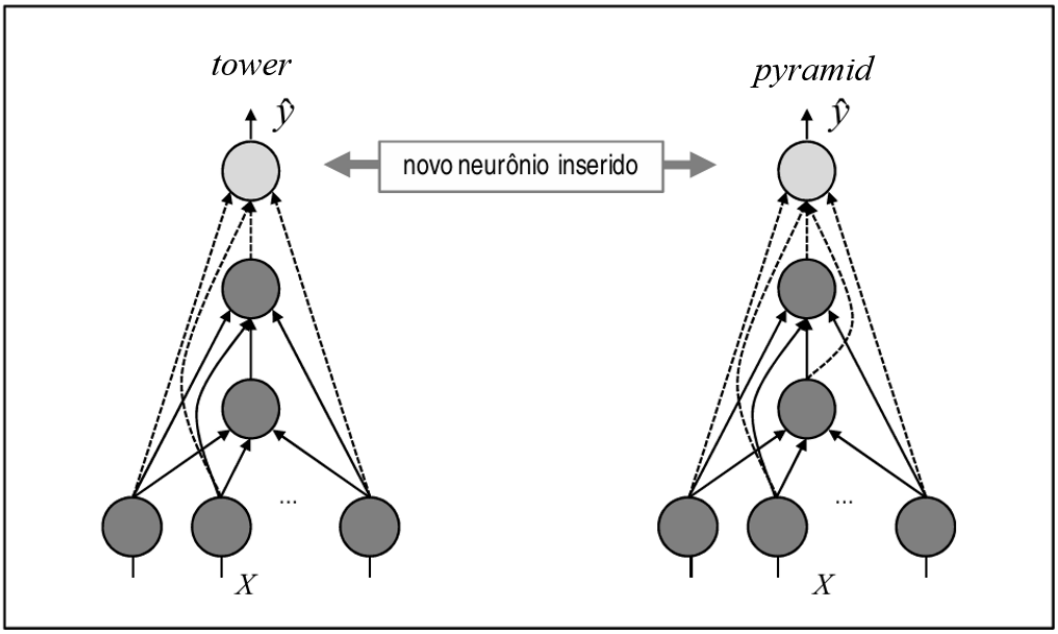
um neurônio de saída associado a problemas de bi-classificação, ou outra de uso mais geral, em forma de cascata com múltiplos neurônios de saída associados a problemas tanto de classificação quanto de regressão de dados.

- Tipo de variáveis de entrada: alguns algoritmos construtivos aceitam apenas entradas binárias, enquanto que outros admitem valores categóricos ordinais e não-ordinais, além de valores numéricos.

2.4.1.1 Algoritmos *tower* e *pyramid*

Entre as contribuições mais importantes de algoritmos construtivos que usam unicamente estratégias de incremento de componentes (conexões e neurônios) na rede, é possível citar os algoritmos tower e pyramid (GALLANT; GALLANT, 1993), concebidos para problemas de classificação de duas classes e com um único neurônio de saída. Esses algoritmos partem de um único neurônio na camada oculta (que é também o neurônio de saída). Cada neurônio inserido na rede é o único neurônio de saída. No *tower*, cada novo neurônio é conectado a todas as entradas e ao último neurônio inserido na rede. Já o *pyramid* difere do *tower* apenas na conectividade do novo neurônio, o qual é conectado com todos os neurônios já inseridos na rede. Por outra parte, ambos utilizam funções de transferência com limiar (*threshold*) e as conexões do novo neurônio são treinadas empregando o algoritmo pocket ratchet modification – PRM (GALLANT *et al.*, 1990).

Figura 13 – Redes *tower* e *pyramid*.



Fonte: Adaptado de Villanueva (2011).

Outro algoritmo com forma distinta de crescimento é o upstart (FREAN, 1990). Este cresce em forma de árvore binária e parte da camada de saída em direção à camada de entrada.

Durante o processo construtivo, um neurônio divide-se em outros dois neurônios visando corrigir o erro cometido pelo neurônio ancestral. Nesta estratégia de correção do erro, os neurônios descendentes têm a tarefa direta de corrigir os erros do neurônio ancestral. Indiretamente, na maioria dos casos, acaba-se por reduzir o erro de classificação da rede neural como um todo.

Considerando as propostas de algoritmos evolutivos que utilizam somente estratégias de poda, tem-se o trabalho de (REED, 1993), o qual descreve algoritmos que partem com uma arquitetura de rede maior do que a necessária e procedem com a remoção de suas conexões e neurônios ocultos. Lahnajärvi e colaboradores (LAHNAJÄRVI *et al.*, 2002) propuseram uma taxonomia para as estratégias de poda: (i) Algoritmos que utilizam estimativas de sensibilidade da função de erro para remover conexões e neurônios ocultos. Aqueles que, ao serem removidos temporariamente produzem menor efeito, são finalmente selecionados para remoção permanente da rede. (ii) Algoritmos munidos com um termo de penalidade adicionado na função-objetivo, de forma a penalizar redes de grande tamanho e que vão contra o princípio da parcimônia. Também podem ser encontradas na literatura propostas de algoritmos que combinam ambas as estratégias (crescimento e poda). Estes tentam tomar vantagem de ambas as estratégias para determinar a arquitetura da rede (FIESLER, 1994; GHOSH; TUMER, 1994). Cabe destacar aqui o algoritmo de aprendizado por busca de projeção (PPL, do inglês projection pursuit learning), o qual produz redes neurais com uma única camada oculta, mas define automaticamente o número de neurônios e o formato de suas funções de ativação, visando maximizar a capacidade de generalização (HWANG *et al.*, 1994; ZUBEN, 1996).

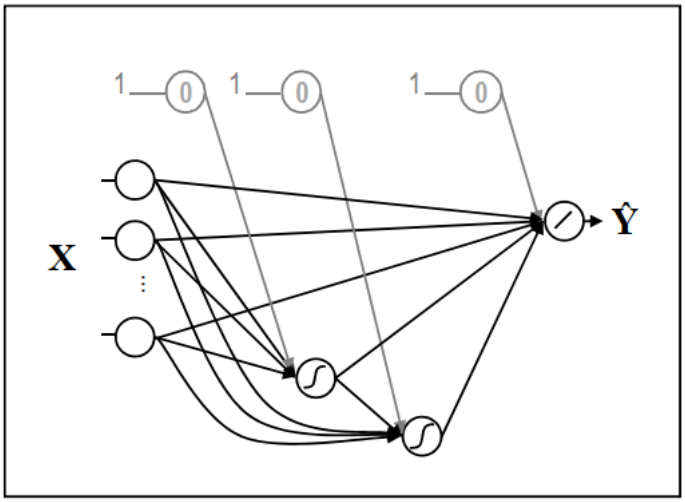
2.4.1.2 O algoritmo construtivo *cascade-correlation* (CasCorr)

O termo CasCorr pode ser atribuído tanto à arquitetura quanto ao algoritmo de aprendizado construtivo. Além de ajustar os pesos sinápticos, também define a topologia da rede neural, em uma forma especial em cascata. O algoritmo começa a partir de uma rede de arquitetura mínima (modelo linear, sem neurônios ocultos) e ajusta otimamente os pesos dos neurônios de saída. Em seguida, adiciona novos neurônios ocultos, um de cada vez, criando uma arquitetura de rede de múltiplas camadas ocultas, com apenas um neurônio por camada, caracterizando a configuração em cascata.

O CasCorr combina duas ideias: (i) a arquitetura em forma de cascada (como ilustrado na Figura 14, considerando apenas dois neurônios ocultos), onde os neurônios ocultos estão dispostos de forma que cada um recebe conexões de todos os neurônios precedentes (inseridos anteriormente e considerando também os da camada de entrada); (ii) o algoritmo de treinamento da rede, que é o responsável pela dinâmica de inserção de novos neurônios (demanda do problema) e do ajuste dos pesos sinápticos.

Em relação aos componentes da arquitetura, a CasCorr é bem parecida à tradicional MLP. Por exemplo, as funções de ativação dos neurônios ocultos são do tipo sigmoidal, seja a função logística ou a tangente hiperbólica. Da mesma forma, os neurônios de saída possuem

Figura 14 – Redes *cascade correlation*.



Fonte: (VILLANUEVA, 2011).

funções lineares. A diferença frente à MLP está: (i) no número de neurônios nas camadas ocultas das redes, sendo que na CasCorr admite-se apenas um neurônio em cada camada oculta, enquanto na MLP este número é arbitrário e pode ser distinto para cada camada; (ii) na disposição das conexões, pois na MLP as conexões são restritas a ligarem neurônios de camadas contíguas, enquanto que na CasCorr cada neurônio recebe todas as conexões possíveis de neurônios em camadas anteriores. Em ambas as redes não se admitem conexões entre neurônios da mesma camada, nem conexões recorrentes. O procedimento de ajuste dos pesos sinápticos é também bem distinto entre ambas as propostas de arquitetura, como já descrito.

2.4.2 Abordagens construtivas com crescimento topológico

As abordagens construtivas baseadas em crescimento topológico diferenciam-se das incrementais por permitirem a modificação dinâmica da estrutura do grafo neural e não apenas a adição sequencial de neurônios. Nesses métodos, a topologia da rede evolui de acordo com a distribuição dos dados de entrada, sendo particularmente adequadas para aprendizado não supervisionado e análise exploratória de dados.

Um exemplo representativo dessa classe é o algoritmo *growing neural gas* (GNG) (FRITZKE, 1995), no qual novos neurônios são inseridos em regiões do espaço de entrada associadas a maior erro de representação. De forma semelhante, mapas auto-organizáveis incrementais permitem o crescimento adaptativo da rede conforme a complexidade dos dados.

Essas abordagens são eficazes na preservação de propriedades topológicas dos dados e na adaptação contínua da rede. Contudo, sua aplicação em tarefas supervisionadas profundas é menos comum, devido à dificuldade de integração com funções de custo complexas e arquiteturas multicamadas.

2.4.3 Abordagens construtivas híbridas: crescimento e poda

As abordagens híbridas combinam estratégias de crescimento estrutural com técnicas de poda, visando equilibrar capacidade de representação e parcimônia. Nesses métodos, a rede neural inicialmente cresce para garantir desempenho adequado e, em seguida, passa por etapas de remoção de neurônios ou conexões redundantes.

Técnicas clássicas de poda incluem o *optimal brain damage* (LECUN *et al.*, 1990) e o *optimal brain surgeon* (HASSIBI; STORK, 1993), que utilizam informações de sensibilidade da função de erro para identificar componentes menos relevantes da rede. Outras abordagens incorporam termos de penalização na função-objetivo, desestimulando arquiteturas excessivamente complexas (REED, 1993).

Embora essas estratégias apresentem bom desempenho em diversos cenários, sua principal limitação reside no aumento do custo computacional, uma vez que o processo de treinamento envolve múltiplas fases distintas.

2.4.4 Abordagens construtivas modulares e relação com *deep learning*

As abordagens construtivas modulares representam uma extensão natural dos algoritmos clássicos de crescimento estrutural, na qual a expansão da rede ocorre por meio da inserção de camadas ou blocos funcionais completos, em vez da adição de neurônios isolados. Essa estratégia favorece a construção de representações hierárquicas profundas, característica central das arquiteturas de *deep learning* (BENGIO, 2009).

No contexto contemporâneo de aprendizado profundo, métodos de crescimento modular têm sido explorados em cenários de aprendizado contínuo e incremental. Por exemplo, arquiteturas dinâmicas modulares que crescem à medida que novas classes ou tarefas são aprendidas têm sido propostas para mitigar o esquecimento catastrófico, mantendo módulos independentes que podem ser adicionados ou reorganizados conforme necessário (TURNER *et al.*, 2021). Esses métodos aproximam-se do princípio construtivo clássico de adaptação automática da estrutura, com a diferença de operarem em níveis de módulo ou sub-rede.

Além disso, a integração de técnicas de *neural architecture search* (NAS) com cenários incrementais evidencia a relevância de adaptações automáticas de arquitetura em redes profundas. Abordagens como a busca de arquitetura adaptativa para aprendizado incremental exploram expansões arquiteturais otimizadas em cada etapa de tarefa, reutilizando pesos e adicionando componentes conforme necessário (HUANG *et al.*, 2019; GAO *et al.*, 2020). Esses métodos compartilham com as abordagens construtivas modulares o princípio fundamental de ajustar dinamicamente a estrutura para melhor acomodar novas informações.

Estruturas modulares e progressivas também aparecem em frameworks de aprendizado contínuo profundo, nos quais a capacidade do modelo é aumentada de forma controlada com a introdução de novos parâmetros que aproveitam representações já aprendidas, preservando

o desempenho nas tarefas anteriores (RUSU *et al.*, 2016; XU *et al.*, 2020). Tais estratégias reforçam a ligação conceitual entre o crescimento modular construtivo e as necessidades de escalabilidade e retenção de conhecimento em *deep learning* moderno.

Dessa forma, as abordagens construtivas modulares estabelecem uma ponte conceitual e prática entre os algoritmos construtivos clássicos e as arquiteturas profundas contemporâneas. Elas mantêm o princípio de adaptação automática da arquitetura à complexidade do problema, ao mesmo tempo em que incorporam mecanismos que atendem às exigências atuais de modularidade, escalabilidade e robustez encontradas em cenários de aprendizado profundo e contínuo.

À luz das abordagens construtivas discutidas, observa-se que os algoritmos clássicos fundamentam-se, em sua maioria, na adição incremental de neurônios individuais ou na modificação gradual da topologia da rede, buscando ajustar a complexidade estrutural ao problema tratado. Embora esses métodos apresentem vantagens como parcimônia e adaptação automática da arquitetura, sua aplicação direta em cenários contemporâneos de aprendizado profundo é limitada, principalmente em razão da baixa escalabilidade, tempo de desenvolvimento e da dificuldade de integração com arquiteturas multicamadas complexas.

Nesse contexto, o algoritmo proposto neste trabalho apresenta os princípios fundamentais do aprendizado construtivo: crescimento progressivo da arquitetura, ajuste estrutural orientado pelo desempenho e treinamento incremental. Entretanto, este trabalho estende essas ideias ao nível de camadas e módulos funcionais, dado que, em vez da inserção de unidades de neurônios, a expansão estrutural ocorre por meio da inserção colaborativa de camadas em redes *stacked autoencoders*, em que cada nova camada ou módulo é treinado de forma isolada, mas alinhado com os blocos já existentes, permitindo que múltiplas partes da rede evoluam simultaneamente sem comprometer as representações previamente aprendidas.

3 Método Colaborativo de Inserção de Camadas

Este capítulo apresenta o princípio e a importância do crescimento estrutural controlado em redes neurais artificiais. O objetivo é descrever como a expansão progressiva da arquitetura, realizada de forma planejada, contribui para o equilíbrio entre aprendizado eficiente, estabilidade e capacidade de generalização do modelo. São discutidos os fundamentos teóricos, os benefícios práticos e o impacto dessa abordagem na formação de representações mais robustas e adaptativas.

O crescimento estrutural controlado de redes neurais é essencial para garantir equilíbrio entre desempenho, estabilidade e generalização. Em metodologias construtivas, a arquitetura da rede não é totalmente definida antes do treinamento. Ela evolui gradualmente, conforme a necessidade de representar padrões mais complexos nos dados. Esse processo permite expandir a capacidade da rede de maneira adaptativa, evitando tanto a limitação de modelos subdimensionados quanto o sobreajuste associado a arquiteturas excessivamente grandes.

A inserção de novas camadas e neurônios deve ocorrer de forma planejada, garantindo que o modelo adapte-se progressivamente sem comprometer o aprendizado já obtido. Essa abordagem promove um treinamento mais estável e cumulativo, em que as camadas iniciais consolidam representações fundamentais antes da inclusão de novos elementos estruturais. Além disso, o controle do crescimento permite aplicar diferentes estratégias de expansão (definidas nesse trabalho como constante, crescente, decrescente e aleatória), ajustando a arquitetura às características específicas do problema e do conjunto de dados.

Do ponto de vista computacional, o crescimento controlado traz benefícios significativos. Iniciar o treinamento com uma rede menor reduz o tempo de processamento e o custo de otimização, já que há menos parâmetros a ajustar nas fases iniciais. À medida que a rede é expandida, o treinamento concentra-se nas novas partes adicionadas, mantendo a eficiência e evitando o retrabalho sobre toda a estrutura. Essa abordagem facilita também o monitoramento da evolução do desempenho da rede em cada etapa de expansão.

Outro ponto relevante é que o crescimento estrutural controlado permite analisar detalhadamente o comportamento do modelo em diferentes níveis de profundidade. Isso possibilita identificar com precisão o ponto em que a adição de novas camadas deixa de trazer ganhos significativos, evitando um crescimento desnecessário da rede. Além disso, ele favorece a interpretação dos resultados, já que a evolução do aprendizado pode ser acompanhada e correlacionada diretamente às mudanças estruturais realizadas.

Em síntese, o crescimento estrutural controlado é um componente fundamental no

desenvolvimento de modelos construtivos. Ele combina **adaptação progressiva, eficiência e robustez**, garantindo que a rede neural evolua de forma coerente com a complexidade dos dados e mantenha um equilíbrio sólido entre capacidade de aprendizado e generalização.

3.1 Uso de Stacked Autoencoders

O uso de *stacked autoencoders* (SAE) é justificado pela sua capacidade de aprendizado profundo e extração hierárquica de características. Cada camada oculta aprende representações progressivamente mais abstratas dos dados de entrada, combinando padrões simples para formar estruturas mais complexas nas camadas subsequentes. O pré-treinamento camada a camada reduz problemas típicos de redes profundas, como o sumiço do gradiente, garantindo um treinamento mais estável.

No contexto do método colaborativo proposto, os decodificadores foram intencionalmente removidos. Essa decisão decorre do fato de que o foco do modelo não está na reconstrução dos dados de entrada, mas na geração direta de representações latentes úteis e discriminantes. Assim, cada codificador atua como um extrator de características autônomo, responsável por produzir representações relevantes que alimentam a próxima camada e contribuem de forma colaborativa para o aprendizado global da rede.

A ausência dos decodificadores reduz a complexidade do modelo e elimina a etapa de reconstrução, que, em muitos casos, não contribui de forma significativa para tarefas supervisionadas, como classificação ou regressão. Diversos estudos apontam que a etapa de decodificação pode inclusive reforçar variações irrelevantes aos padrões discriminantes, sem ganho efetivo no desempenho (ALBERTI *et al.*, 2017; MAJUMDAR; TRIPATHI, 2017).

Embora essa remoção tenha sido adotada no modelo proposto, o método desenvolvido mantém a possibilidade de ser estendido para tarefas de reconstrução, caso necessário, mediante a incorporação de mecanismos apropriados que permitam recuperar a informação de entrada.

3.1.1 Limitações dos autoencoders

Apesar da capacidade dos *autoencoders* de extrair representações significativas, seu treinamento apresenta desafios, especialmente em redes profundas:

- Sumiço do gradiente (*vanishing gradient*): dificulta que as camadas iniciais aprendam representações relevantes.
- Explosão do gradiente (*exploding gradient*): pode causar atualizações de pesos excessivas e instabilidade.
- *Overfitting* ou memorização: redes muito grandes podem memorizar os dados em vez de aprender representações úteis.

- Dificuldade em capturar características complexas: camadas profundas podem aprender representações superficiais.
- Sensibilidade a hiperparâmetros: pequenas variações em taxa de aprendizado, função de ativação ou número de neurônios podem comprometer a convergência.
- Distanciamento do erro verdadeiro: a inserção de novas camadas pode alterar a dinâmica da rede, prejudicando a aproximação do valor desejado.

Dessa forma, o SAE empregado neste trabalho atua essencialmente como um empilhamento de codificadores progressivos, alinhando-se ao princípio do método colaborativo: permitir a expansão gradual da rede, camada a camada, mantendo o conhecimento adquirido e otimizando continuamente a geração de novas representações, melhorando a estabilidade e a robustez da rede.

3.2 Método Proposto

Redes neurais artificiais (RNA) com múltiplas camadas ocultas apresentam, durante o treinamento, problemas relacionados ao *desvanecimento do gradiente* e ao distanciamento do erro verdadeiro. Nesse contexto, a inserção de uma camada por vez seria o ideal, permitindo um aprendizado mais controlado e gradual. Entretanto, a adição de novas camadas pode interferir negativamente no conhecimento previamente adquirido, sendo necessária uma estratégia que preserve o aprendizado já obtido.

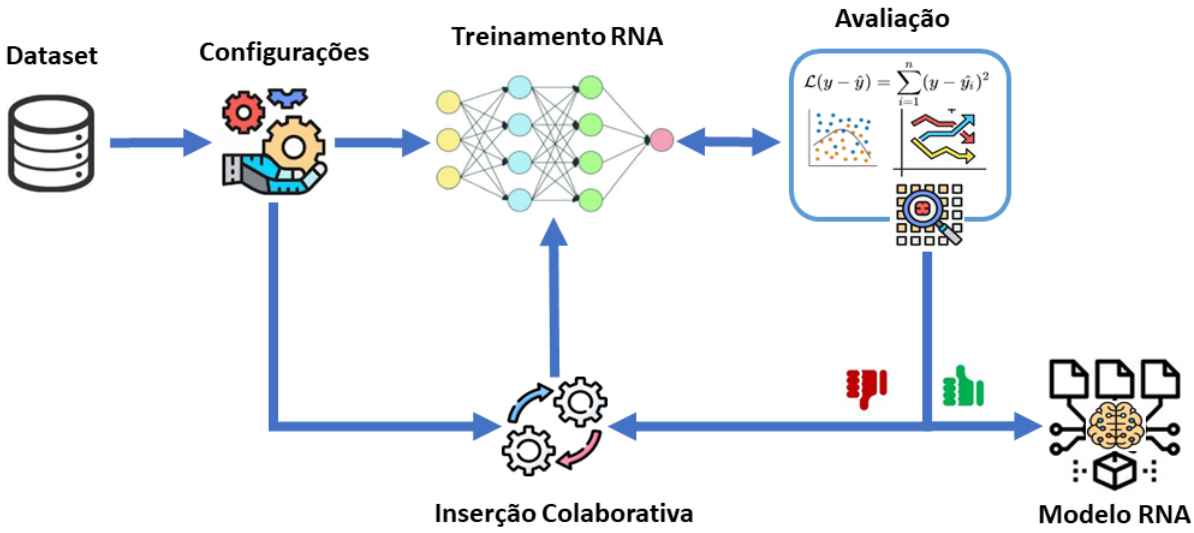
Com base nessa premissa, o presente trabalho propõe uma metodologia para a inserção de novas camadas ocultas em redes do tipo *stacked* autoencoder, fundamentada no conceito de ramo. De forma conceitual, Um ramo é uma sequência de camadas que constitui um caminho funcional independente, capaz de aprender representações próprias. A partir desse conceito, introduz-se um método inovador caracterizado como **colaborativo** (SAE CollabNet), no qual a adição de novos ramos não substitui o aprendizado anterior, mas atua de forma complementar, colaborando com os ramos já existentes. Ou seja, os ramos anteriores permanecem inalterados, enquanto as novas camadas ajustam suas ativações para complementar e refinar a saída global da rede. Dessa forma, os diferentes ramos "colaboram" entre si, combinando suas representações de maneira sinérgica e garantindo que o conhecimento previamente adquirido seja preservado.

3.2.1 Inserção colaborativa

O sistema de inserção colaborativa é ilustrado na Figura 15. Inicialmente, os dados são tratados e depois divididos em conjuntos de treinamento, validação e teste. Em seguida, são configurados os parâmetros da rede e do algoritmo de treino, incluindo o número de neurônios

por camada, a taxa de aprendizado e o número de épocas de treinamento. A rede é então treinada utilizando o algoritmo de *backpropagation*. Após o treinamento, o desempenho da rede é avaliado: se satisfatório, obtém-se um modelo final para o problema; caso contrário, inicia-se o processo de inserção colaborativa, repetindo o ciclo a partir da etapa de treinamento da rede, de modo a incrementar progressivamente a capacidade de aprendizado sem comprometer o conhecimento já adquirido.

Figura 15 – Sistema de inserção colaborativa.

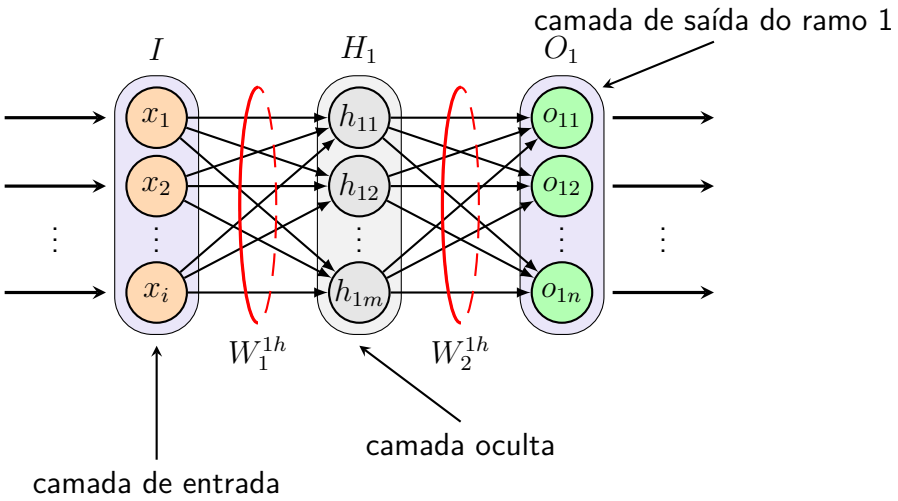


Fonte: Elaborada pelo autor.

O método de inserção colaborativa é estruturado em três etapas distintas. Na etapa 1, conforme ilustrado na Figura 16, é inicializada uma rede neural simples composta por uma camada de entrada, uma camada oculta e uma camada de saída. Essa rede inicial é treinada de forma convencional e passa a ser referida como ramo 1. O treinamento do ramo 1 estabelece a base do conhecimento da rede, capturando padrões iniciais dos dados de entrada e definindo representações intermediárias que servirão como referência para as etapas subsequentes de expansão e integração colaborativa de novas camadas.

De acordo com a Fig. 16, a arquitetura do ramo 1 é apresentada composta por três camadas sequenciais: entrada, oculta e saída. A camada de entrada é representada por I, a camada oculta simbolizada por H_1 e a camada de saída é denotada por O_1. Com relação às unidades internas das camadas, a camada de entrada é composta por i neurônios (cor laranja) formando um vetor $X = (x_1, x_2, \ldots, x_i)$, a camada oculta é composta por m neurônios de cor cinza que geram um vetor $H_1 = (h_{11}, h_{12}, \ldots, h_{1m})$ na saída dessa camada e a camada de saída é composta por n neurônios de cor verde e que resultam em um vetor $O_1 = (o_{11}, o_{12}, \ldots, o_{1n})$. A ponderação dos dados que trafegam entre as camadas é feita por meio de pesos, onde W_1^{1h} representa a ponderação das ligações dos neurônios entre a camada de entrada e a camada oculta, enquanto que W_2^{1h} representa a ponderação entre os neurônios da camada oculta e da camada de saída.

Figura 16 – Estrutura de uma rede neural com configuração mínima (ramo 1).



Fonte: Elaborada pelo autor.

A Equação 3.1 define o processo que resulta nos dados na saída da camada oculta H_1 . Nessa equação, os dados X passam por uma combinação através da ponderação W_1^{1h} e um viés b_1^{1h}. Em seguida, a função de ativação f processa esse resultado e gera um vetor de saída. A função de ativação da camada oculta é definida pelo projetista, permitindo flexibilidade na escolha da não-linearidade mais adequada para extrair representações complexas dos dados de entrada

$$H_1 = f(W_1^{1h} X + b_1^{1h}) \tag{3.1}$$

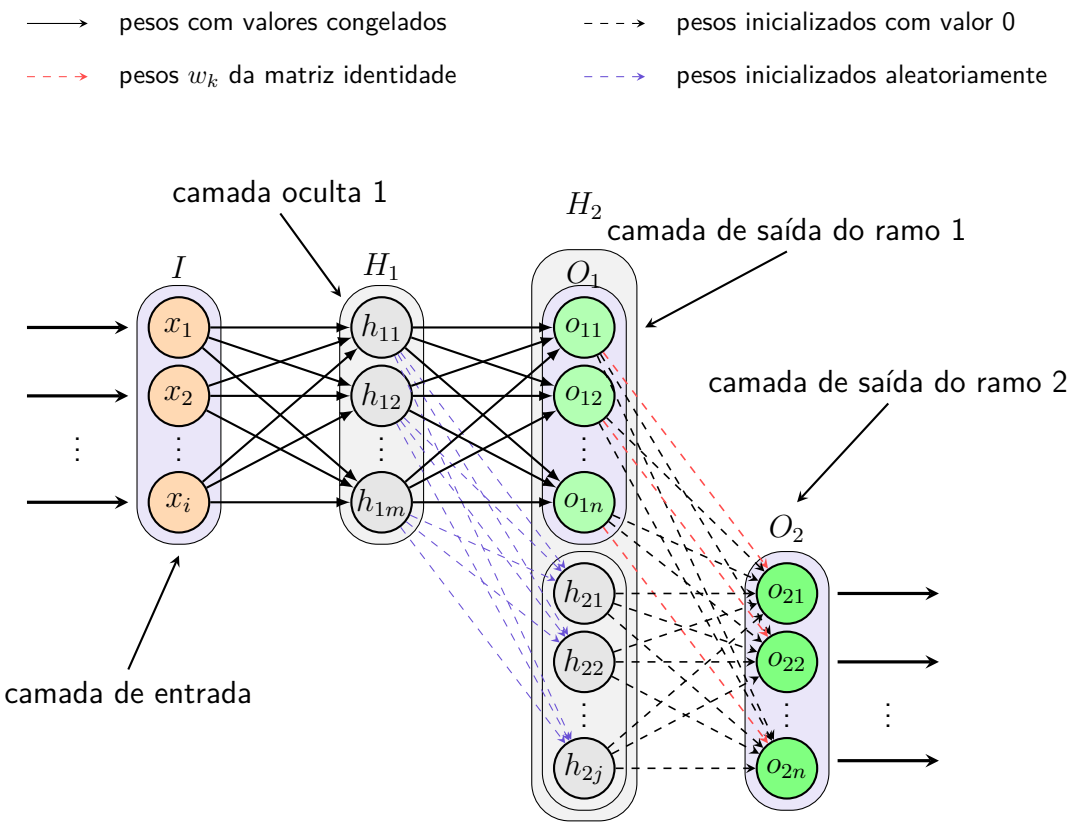
A Equação 3.2 define a função de ativação dos neurônios da camada de saída. Nessa equação, W_2^{1h} representa os pesos entre a camada oculta e a camada de saída, e b_2^{1h} é o viés da camada de saída. O neurônio da camada de saída possui função de ativação g do tipo linear, garantindo que os valores de saída mantenham a escala apropriada para a tarefa de predição ou regressão.

$$O_1 = g(W_2^{1h} H_1 + b_2^{1h}) \tag{3.2}$$

Na etapa 2, ilustrada na Figura 17, é criado o ramo 2, composto por uma nova camada oculta e uma nova camada de saída. Os dados de entrada para este novo ramo provêm da saída da camada oculta do ramo 1, representada por H_1. Dessa forma, o ramo 2 recebe as representações intermediárias já extraídas pela rede inicial, permitindo que novas camadas construam sobre informações previamente processadas. Além disso, a saída do ramo 1, denotada pelo vetor O_1, é conectada à camada de saída do ramo 2 por meio de pesos de influência w_k. Essa conexão regula o grau de contribuição das ativações já aprendidas pelo ramo 1, equilibrando a preservação do conhecimento antigo com a integração das novas representações

no ramo 2. Além disso, os pesos entre as camadas ocultas e de saída do ramo 2 são inicializados nulos. Essa configuração dos pesos conectados à camada de saída do ramo 2 possibilita um aprendizado colaborativo entre os ramos, pois preserva a saída já aprendida, mas possibilita ampliar a capacidade da rede de explorar o espaço de busca sem comprometer informações previamente adquiridas.

Figura 17 – Processo de inserção colaborativa de uma nova camada.



Fonte: Elaborada pelo autor.

Conforme definido anteriormente, a entrada principal do ramo 2 é a saída da camada oculta do ramo 1, permitindo que o novo ramo construa sobre as representações já aprendidas. Com a inserção de uma nova camada oculta via ramo 2, o modelo passa a explorar uma nova região do espaço de dados, promovendo uma alteração na geometria da superfície de erro e criando trajetórias alternativas de descida do gradiente. Com isso, consegue-se favorecer a fuga de mínimos locais associados à arquitetura anterior, sem comprometer o conhecimento já consolidado.

A camada de saída do ramo 2 possui conexões diretas com os neurônios de sua própria camada oculta, representadas pelos pesos w_2^{2h}, bem como conexões com a saída do ramo anterior, representadas pela matriz de pesos $I_k(n)$. Esses dois componentes são concatenados para formar a matriz de pesos total do ramo 2, W_2^{2h}, conforme definido na Equação 3.3. Inicialmente, os pesos w_2^{2h} são inicializados com valor zero, garantindo que o novo ramo não perturbe a saída já aprendida pelo ramo 1. Essa configuração permite que o ramo 2 seja treinado

de maneira estável, ajustando seus pesos por meio do *backpropagation* para complementar a saída do ramo 1.

Durante o treinamento do ramo 2, o ramo 1 permanece fixo, ou seja, seus pesos previamente treinados não são alterados. Como a saída do ramo 1 é incorporada como entrada adicional na camada oculta do ramo 2, a saída final da rede torna-se a soma das saídas dos dois ramos. A conexão $I_k(n)$ é definida como uma matriz diagonal, em que os elementos da diagonal principal possuem inicialmente o valor $k = 1$, garantindo que a influência da saída do ramo 1 seja transmitida integralmente ao ramo 2.

$$W_2^{2h} = [I_k(n)\ w_2^{2h}] = \begin{bmatrix} k & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & k & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{3.3}$$

Com essas definições, a saída da camada oculta do ramo 2, denotada por H_2, é expressa pela Equação 3.4, onde W_1^{2h} representa a matriz de pesos entre a camada oculta e a camada de saída do ramo e f, a função de ativação do tipo não linear.

$$H_2 = \begin{bmatrix} O_1 \\ f(W_1^{2h} H_1 + b_1^{2h}) \end{bmatrix} \tag{3.4}$$

A saída do ramo 2 é definida pela Equação 3.5, em que b_2^{2h} é o vetor de *bias* correspondente.

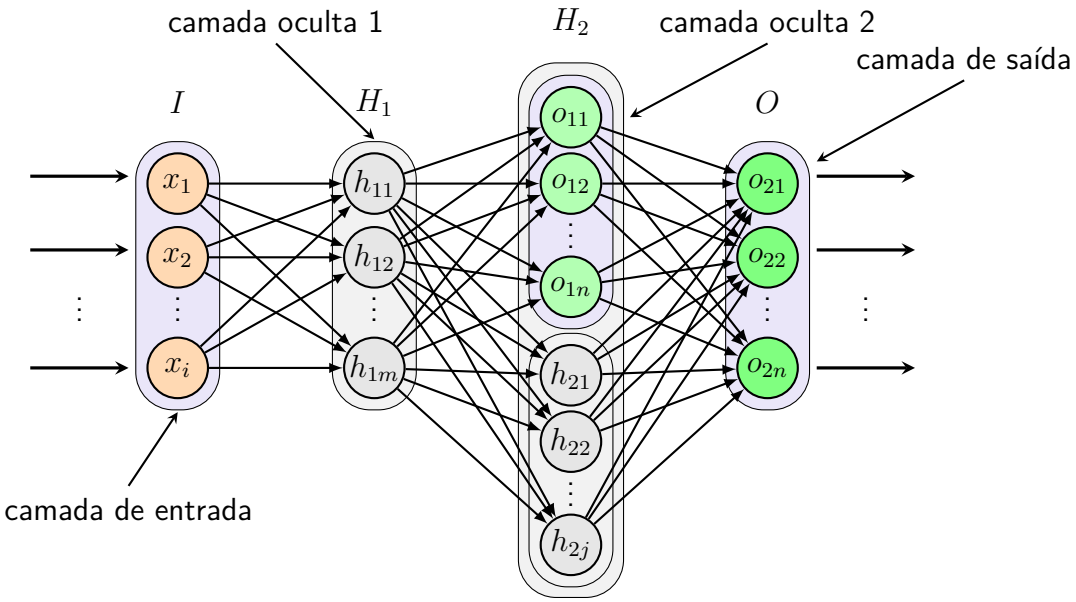
$$O_2 = g(W_2^{2h} H_2 + b_2^{2h}) \tag{3.5}$$

Finalmente, na etapa 3, a saída do *ramo 2* é ajustada para representar a saída global da rede, incorporando a contribuição de todos os ramos anteriores. Nesse processo, a saída do ramo 1 é utilizada como uma das entradas da camada de saída do ramo 2, permitindo que as informações previamente aprendidas sejam preservadas e combinadas com as novas ativações. Com a conclusão dessa etapa, a rede adquire a seguinte estrutura final: uma camada de entrada, duas camadas ocultas (uma de cada ramo) e uma camada de saída que representa a combinação das saídas individuais. A Fig. 18 ilustra essa configuração.

O processo de inserção colaborativa é interrompido quando uma intervenção do projetista é realizada ou quando o algoritmo de colaboração atinge uma condição de finalização pré-definida, garantindo que a expansão da rede ocorra de maneira controlada e estável.

Para ampliar o espaço de busca e mitigar o efeito de sobreponderação da entrada decorrente da inserção de novas camadas ocultas, introduz-se um *ramo extra* composto exclusivamente por neurônios ocultos, compondo novos elementos na camada escondida do

Figura 18 – Rede com duas camadas ocultas criada pela inserção colaborativa.



Fonte: Elaborada pelo autor.

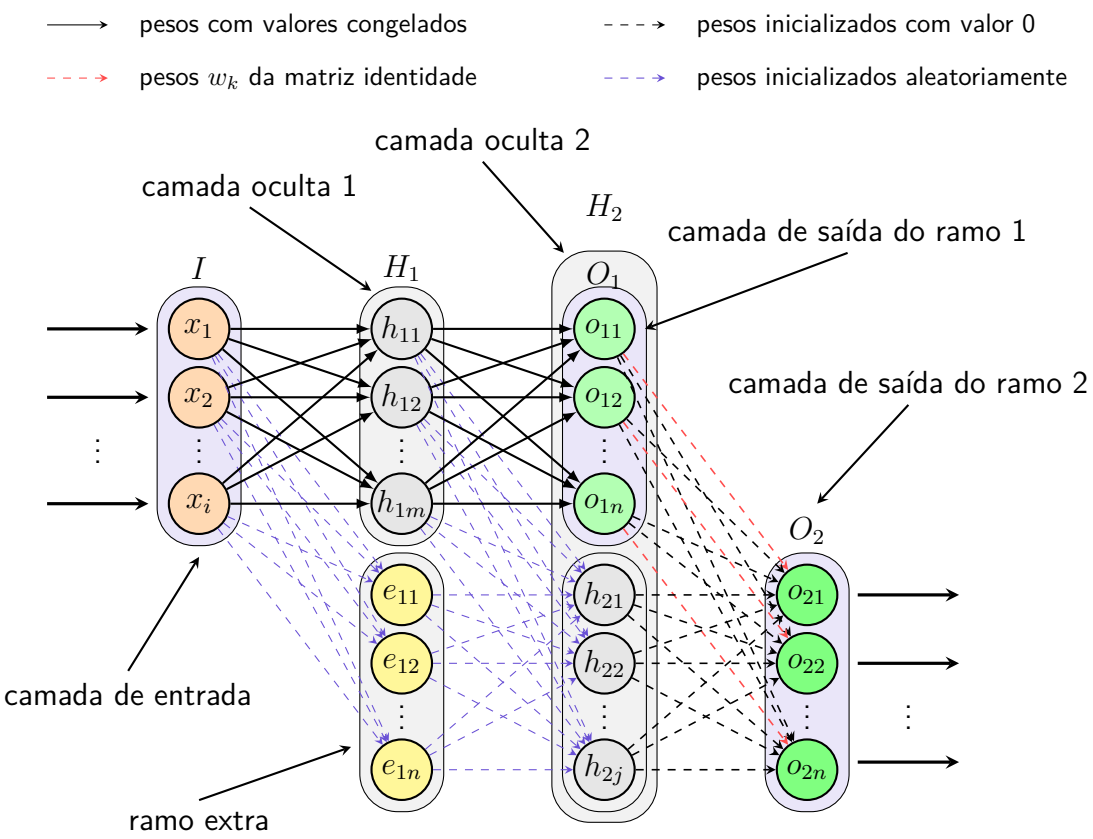
ramo anterior. Esse ramo estabelece uma conexão direta entre a camada de entrada original e a camada oculta do ramo 2, fornecendo informações não processadas que complementam as ativações já abstraídas pelas camadas anteriores. O vetor de ativações do ramo extra é representando por $E = (e_1, e_2, \ldots, e_n)$.

A Figura 19 ilustra essa configuração. No diagrama, o ramo extra é destacado pelo neurônio de cor amarela, evidenciando seu caminho funcional independente, que conduz os dados brutos diretamente ao ramo 2, sem atravessar as transformações do ramo 1. Ao preservar componentes do sinal original e integrá-los às representações de nível mais alto, o ramo extra amplia a diversidade representacional da rede. Essa combinação entre informações de baixo e alto nível favorece maior flexibilidade no aprendizado, aumenta a capacidade de adaptação e contribui para a estabilidade do processo de otimização.

A inserção colaborativa mostra-se uma estratégia eficiente para o crescimento estrutural de redes neurais, pois a adição de novos ramos ocorre sem que seja necessário submeter toda a arquitetura a um novo ciclo completo de treinamento. Essa característica reduz o custo computacional do processo de expansão, uma vez que restringe a atualização dos parâmetros ao bloco recém-adicionado. De modo prático, essa dinâmica proporciona uma redução do tempo de projeto da RNA, uma vez que o treinamento, além de exigir menor esforço computacional, passa a ter a seleção de hiperparâmetros de forma mais flexível, pois minimiza a necessidade de ajustes exaustivos nos hiperparâmetros e na configuração estrutural da rede. Nesta proposta, consegue-se melhorar o aprendizado, mesmo com os hiperparâmetros não tendo necessariamente os valores mais adequados.

Um efeito adicional do ganho de tempo no projeto completo da rede reside no fato

Figura 19 – Processo de inserção colaborativa de uma nova camada com adição de ramo extra.



Fonte: Elaborada pelo autor.

de que, com menos ajustes nos hiperparâmetros, menos processamentos são realizados e isso reflete diretamente em menor consumo de energia, aspecto desejável em experimentos de larga escala e em cenários de escalabilidade.

Na etapa de construção da rede proposta, foram analisadas diferentes estratégias de crescimento para o *stacked autoencoder colaborativo*. Essas estratégias determinam como novas camadas ocultas são inseridas e conectadas às camadas já existentes, influenciando diretamente a capacidade de representação e o custo computacional da rede.

O crescimento estrutural da rede foi cuidadosamente controlado considerando dois fatores principais:

1. Número de neurônios adicionados em cada nova camada oculta: esse parâmetro define como a capacidade da rede é expandida à medida que novas camadas são inseridas. A adição de neurônios pode seguir diferentes tendências:

   - Constante (CNT): a mesma quantidade de neurônios é adicionada em cada nova camada;
   - Crescente (CRT): o número de neurônios aumenta progressivamente nas camadas subsequentes;

- Decrescente (DRT): o número de neurônios diminui nas camadas posteriores;
- Aleatória (ALT): o número de neurônios é definido de forma randômica, permitindo maior diversidade estrutural.

Na Figura 20, diferentes estratégias de crescimento de camadas e de inserção de ramos extras são ilustradas. Observa-se que, independentemente do tipo de inserção (constante, crescente, decrescente ou aleatória), os novos ramos (representados por nós e conexões em cores diferenciadas) recebem informações tanto da camada de saída antiga quanto das camadas ocultas intermediárias já existentes. Essa interconexão assegura que a nova estrutura aprenda complementando e ajustando o conhecimento já adquirido, caracterizando o processo de aprendizado colaborativo entre os ramos da rede.

O crescimento estrutural controlado da rede neural tem papel fundamental no equilíbrio entre desempenho, estabilidade e capacidade de generalização do modelo. Em abordagens construtivas, a rede não é totalmente definida no início do treinamento; ao contrário, sua arquitetura evolui gradualmente conforme a necessidade de representar padrões mais complexos nos dados. Esse processo permite uma expansão progressiva da capacidade de aprendizado, evitando tanto a limitação de redes subdimensionadas quanto o sobreajuste causado por arquiteturas excessivamente grandes desde o início.

Ao adicionar novas camadas ou neurônios de forma controlada, o modelo é capaz de adaptar sua estrutura à medida que a tarefa exige maior poder de representação. Essa adaptação gradual contribui para um treinamento mais estável, pois as camadas iniciais aprendem representações básicas e consolidadas antes que novas unidades sejam introduzidas. Além disso, o controle sobre a taxa e a forma de crescimento, seja constante, crescente, decrescente ou aleatória, permite explorar diferentes estratégias de expansão, ajustando a rede às características específicas de cada problema e conjunto de dados.

Outro ponto relevante é a eficiência computacional. Iniciar o treinamento com uma rede pequena reduz o custo de processamento e o tempo de convergência, já que menos parâmetros precisam ser otimizados nas etapas iniciais. À medida que o modelo cresce, o treinamento concentra-se apenas em uma pequena parte da rede, nos pontos em que novas adaptações são necessárias, evitando desperdício de recursos. Essa abordagem propicia a interpretabilidade e o diagnóstico da evolução do aprendizado, uma vez que o comportamento da rede pode ser avaliado a cada fase de expansão.

Em síntese, o crescimento estrutural controlado é uma estratégia essencial em modelos construtivos, pois combina adaptação progressiva, eficiência e robustez, garantindo que a rede evolua em função das demandas reais da tarefa e mantenha um equilíbrio entre capacidade de aprendizado e generalização.

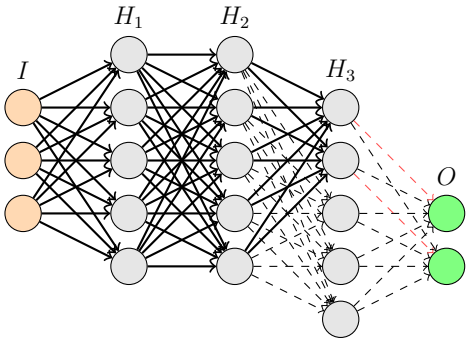
2. Ajuste dos pesos w_k: Este fator controla a conexão entre a nova camada oculta (ou novo ramo) e a camada de saída da arquitetura previamente existente. Os pesos w_k definem o

quanto as ativações da camada recém-adicionada influenciam as saídas já aprendidas pela rede, sendo determinante para o equilíbrio entre preservação do conhecimento antigo e integração de novas representações.

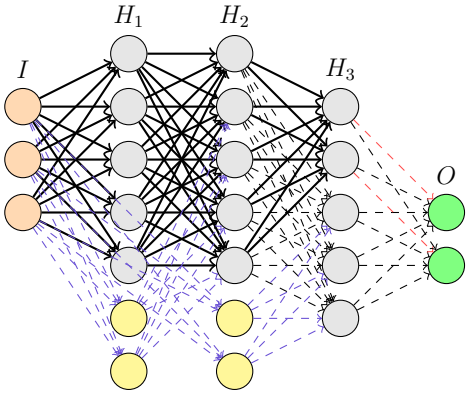
Com base na combinação dessas duas dimensões, foram definidas quatro configurações experimentais distintas do método, descritas a seguir:

- M1: pesos w_k constante e sem ramo extra. Nessa configuração, a rede cresce de forma controlada sem adicionar caminhos paralelos, mantendo a influência das novas camadas uniforme sobre a saída existente.
- M2: pesos w_k ajustável e sem ramo extra. Aqui, a rede permite que a influência das novas camadas varie dinamicamente durante o treinamento, mas ainda sem introduzir ramos adicionais, favorecendo adaptação gradual das ativações.
- M3: pesos w_k constante com ramo extra. Nessa configuração, além da adição de camadas, um ramo paralelo é incluído, mantendo os pesos w_k fixos. O ramo extra possibilita a introdução de novas representações sem modificar diretamente a trajetória das camadas originais.
- M4: pesos w_k ajustável com ramo extra. Combina a flexibilidade do ajuste dinâmico dos pesos w_k com a inserção de um ramo paralelo, permitindo que as novas ativações contribuam de forma adaptativa para a saída da rede, promovendo aprendizado mais robusto e eficiente.

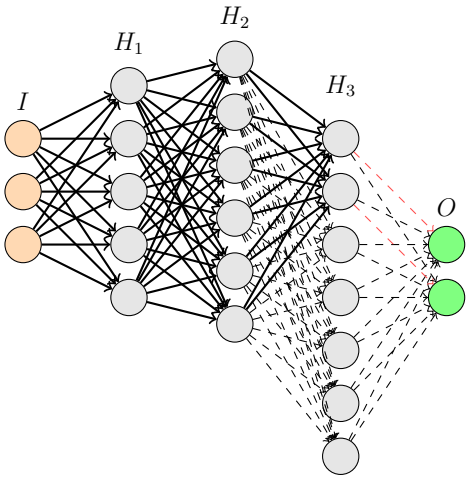
Figura 20 – Modelos desenvolvidos com a SAE CollabNet.



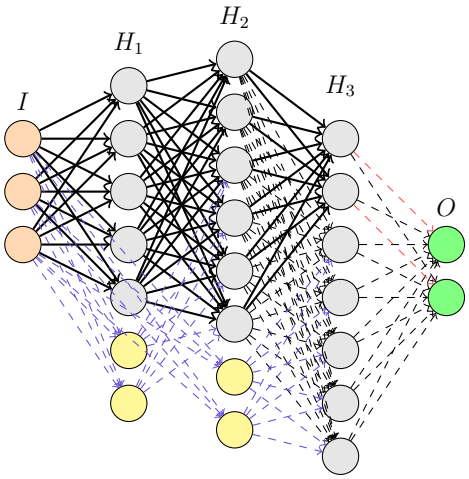
(a) Inserção constante sem ramo extra



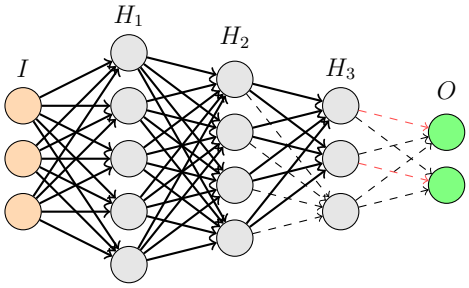
(b) Inserção constante com ramo extra



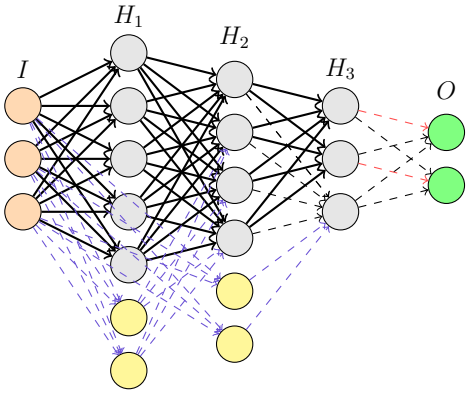
(c) Inserção crescente sem ramo extra



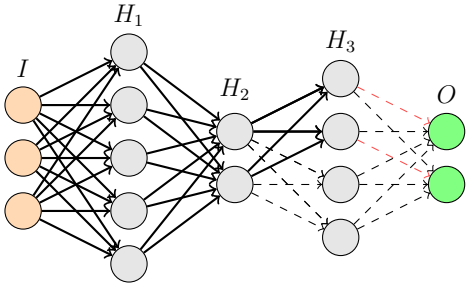
(d) Inserção crescente com ramo extra



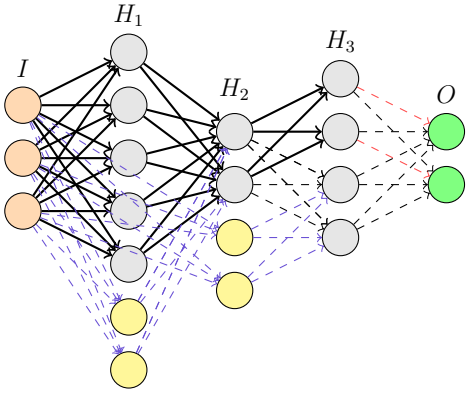
(e) Inserção decrescente sem ramo extra



(f) Inserção descrescente com ramo extra



(g) Inserção aleatória sem ramo extra



(h) Inserção aleatória com ramo extra

Fonte: Elaborada pelo autor.

3.2.2 Função de Custo e Otimização

A função de custo é um componente central no treinamento de redes neurais, pois quantifica a discrepância entre as saídas previstas pela rede e os valores reais do conjunto de dados. No presente trabalho, utiliza-se o erro quadrático médio (*Mean Squared Error*, do inglês - MSE), uma métrica amplamente empregada em redes do tipo *autoencoder* e em problemas de regressão. O MSE é calculado conforme a Eq. 3.6, onde n representa o número de amostras, O_i a saída real e $\hat{O}_i$ a saída predita pela rede. Assim, o MSE permite a otimização dos parâmetros $\{W_1^i, b_1^i, W_2^i, b_2^i\}$ de uma função custo θ_f ao longo do treinamento.

$$J_{MSE}(\theta_f) = \frac{1}{n}\sum_{i=1}^{n} L_{MSE}(O_i, O_i') = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}||O_i - O_i'||^2 \tag{3.6}$$

No contexto de *stacked autoencoders* e do método colaborativo, o MSE oferece diversas vantagens. Primeiramente, ele fornece um sinal contínuo e diferenciável, essencial para o algoritmo backpropagation, permitindo o ajuste eficiente de pesos e *bias* em cada camada. Além disso, o MSE mede diretamente a diferença entre as saídas preditas de cada ramo e os valores reais, garantindo que a inserção de novas camadas contribua de forma complementar à rede existente, sem degradar o conhecimento previamente adquirido e corrigindo eventuais aprendizados erroneos.

Outra vantagem é que o MSE é sensível a grandes erros individuais, penalizando discrepâncias significativas de forma mais intensa. Isso é importante no método colaborativo, pois garante que cada camada inserida seja treinada para corrigir ou refinar representações de maneira eficaz. Entretanto, essa sensibilidade também pode tornar o treinamento mais suscetível a *outliers*, sendo necessário cuidado na preparação e normalização dos dados.

O uso da função de custo MSE, combinado com o crescimento estrutural controlado, possibilita uma otimização gradual e estável. Cada nova camada adicionada à rede é treinada considerando o retorno fornecido pelo MSE, o que permite a incorporação progressiva de conhecimento e a evolução coerente da arquitetura em função da complexidade dos dados. Como resultado, essa abordagem garante que o modelo alcance um equilíbrio entre capacidade de generalização e preservação do conhecimento, além de facilitar a análise do desempenho em cada etapa de expansão da rede.

3.3 Desenvolvimento de uma interface

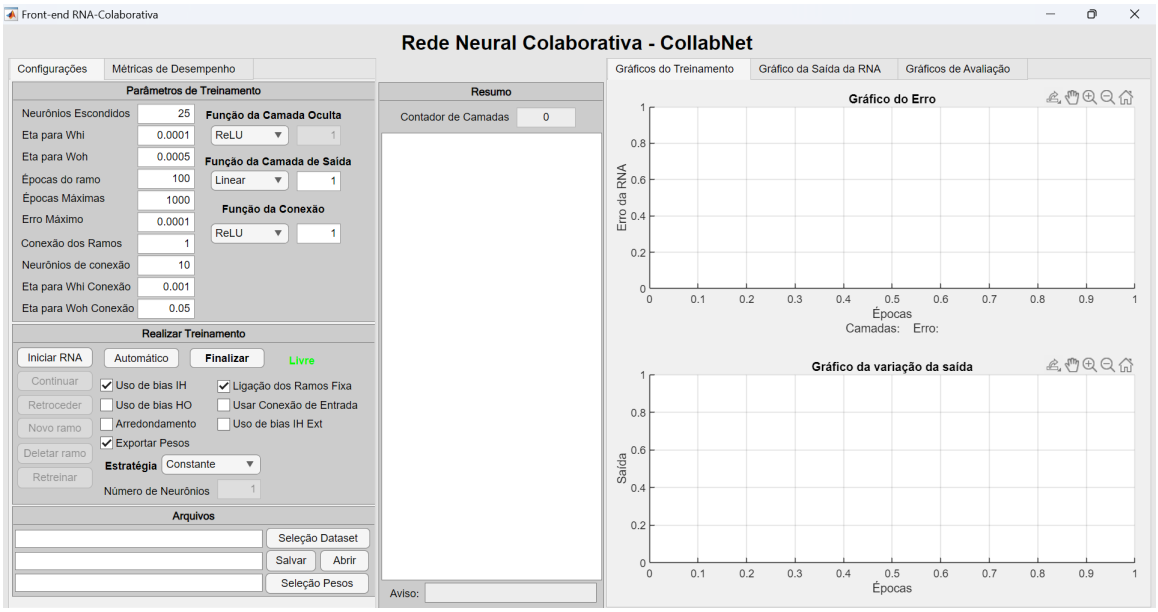
Para viabilizar a configuração, execução de testes e armazenamento dos resultados, foi desenvolvida uma interface no *software* Matlab, ilustrada na Figura 21. Essa interface foi projetada para tornar o uso da metodologia colaborativa mais intuitivo, permitindo que usuários configurem, monitorem e ajustem a rede de maneira eficiente sem a necessidade de interações diretas com o código-fonte.

Através dessa interface, é possível definir diversos parâmetros essenciais para o treinamento e expansão da rede. Entre eles, destacam-se: a quantidade de neurônios a serem adicionados em novas camadas ocultas, permitindo controlar o crescimento estrutural; o número de épocas de treinamento, possibilitando ajustes finos na duração do aprendizado; a função de ativação de cada camada, que influencia a não linearidade das representações aprendidas; a taxa de aprendizado, que determina a velocidade com que os pesos são atualizados; e os métodos de inserção de camadas, que serão detalhados no capítulo seguinte. Além disso, a interface permite habilitar ou desabilitar o uso de *bias*, carregar ou salvar dados de entrada e resultados, criar ou excluir ramos da rede e retomar o treinamento de ramos já existentes, garantindo flexibilidade total no gerenciamento da arquitetura.

Outro aspecto importante da interface é a capacidade de monitoramento em tempo real. Durante o treinamento, é possível visualizar gráficos de evolução das métricas de erro, desempenho em conjunto de teste e validação, além de informações detalhadas sobre o comportamento de cada camada ou ramo inserido. Esse acompanhamento facilita a análise da eficácia das novas camadas, permite detectar problemas de convergência ou sobreajuste precocemente e possibilita ajustes dinâmicos nos parâmetros, tornando o processo de treinamento mais controlado e seguro.

A interface também atua como um ambiente de experimentação para a metodologia colaborativa, integrando todas as funcionalidades necessárias para aplicar, testar e validar os diferentes métodos de inserção de camadas. Ela oferece uma visão consolidada do estado da rede, permitindo comparar resultados entre diferentes configurações, documentar experimentos e gerar relatórios consistentes, o que é fundamental para a reprodutibilidade dos experimentos.

Figura 21 – Interface para treinamento da rede neural colaborativa.



Fonte: Elaborada pelo autor.

No capítulo seguinte, serão mostrados os resultados obtidos com a metodologia colaborativa utilizando a interface desenvolvida, demonstrando como cada funcionalidade contribuiu para a execução dos experimentos e a avaliação da performance da rede.

4 Resultados

Neste capítulo, são apresentadas detalhadamente as bases de dados utilizadas no desenvolvimento deste trabalho, abrangendo diferentes domínios e finalidades experimentais. Foram empregados conjuntos de dados relacionados a diagnóstico de câncer, reconhecimento de padrões em sinais de áudio e os conjuntos da *OpenML-CC18*, estes últimos utilizados em duas frentes distintas: nos experimentos de mutação de função de ativação e nas análises de classificação binária.

Essa diversidade de bases foi fundamental para avaliar o comportamento e a robustez da rede SAE CollabNet em diferentes contextos, desde cenários com dados estruturados e bem definidos, como os de saúde, até situações com maior variabilidade e ruído, como nas tarefas envolvendo áudio.

Além da descrição dos conjuntos de dados, este capítulo apresenta as métricas de avaliação adotadas para medir o desempenho dos modelos e discute os resultados obtidos a partir dos experimentos conduzidos. A combinação entre múltiplas bases e diferentes tipos de tarefas permitiu validar a generalização da abordagem proposta e evidenciar suas vantagens frente a métodos tradicionais de redes neurais construtivas.

4.1 Bases de Dados de Câncer

Para o treinamento da rede neural, foi considerada uma tarefa de classificação de padrões. Para tal, utilizou-se a base de dados *Breast Cancer Wisconsin* (WOLBERG, 1992), disponível no repositório de aprendizado de máquina da Universidade da Califórnia em Irvine (UCI). Essa base é amplamente utilizada em pesquisas de aprendizado de máquina devido à sua diversidade e à representação equilibrada de classes, permitindo avaliar a capacidade de generalização de modelos de classificação.

A base contém nove atributos que descrevem características morfológicas das células tumorais e uma variável de classificação, totalizando dez variáveis. As características são detalhadas a seguir:

- Espessura da massa celular (CT);
- Uniformidade do tamanho das células (CS);
- Uniformidade do formato das células (CH);
- Adesão marginal (AD);
- Tamanho das células epiteliais (EP);

- Núcleo vazio (BN);
- Cromatina branda (CO);
- Nucléolo Normal (NN);
- Mitose (MM);
- Classificação do tumor ("benigno" ou "maligno").

A base é composta por 699 amostras, sendo aproximadamente 65% referentes a tumores benignos e 35% a tumores malignos. Para o treinamento da rede neural, os dados foram divididos em 70% para treinamento e 15% para validação e testes, respectivamente. Essa divisão permite tanto o ajuste dos parâmetros da rede quanto a avaliação imparcial da capacidade do modelo em generalizar para dados não vistos durante o treinamento.

4.1.1 Métricas de Avaliação

A avaliação do desempenho da rede neural foi realizada por meio do erro quadrático médio (MSE) e da acurácia. O MSE, já detalhado no capítulo de Metodologia, permite quantificar o erro médio entre os valores preditos pelo modelo e os valores reais, sendo uma métrica essencial para o ajuste das camadas e parâmetros da rede.

A acurácia fornece uma medida direta da proporção de classificações corretas realizadas pelo modelo, permitindo avaliar a eficácia da rede em tarefas de classificação binária. Ela é calculada conforme a Equação 4.1:

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.1}$$

onde:

- Verdadeiro positivo (VP): o modelo prevê a classe positiva e a classe real é positiva;
- Verdadeiro negativo (VN): o modelo prevê a classe negativa e a classe real é negativa;
- Falso positivo (FP): o modelo prevê a classe positiva, mas a classe real é negativa;
- Falso negativo (FN): o modelo prevê a classe negativa, mas a classe real é positiva.

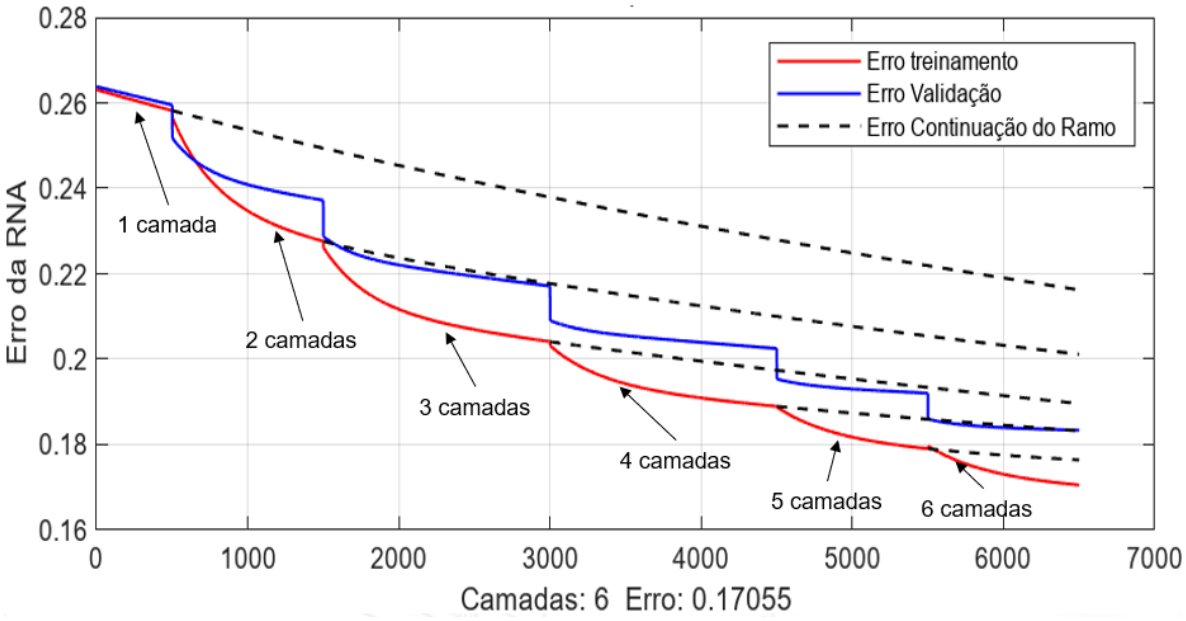
O uso combinado de MSE e acurácia possibilita analisar tanto a exatidão das predições numéricas quanto a capacidade da rede de classificar corretamente cada amostra. Essas métricas permitiram avaliar o desempenho das camadas inseridas e a qualidade geral do modelo, cujos resultados serão apresentados na seção seguinte.

4.1.2 Resultados dos Experimentos

Com a aplicação da metodologia proposta, foi projetada uma rede neural, cujo esquema está ilustrado na Figura 22. Nesse gráfico, a linha preta tracejada representa a continuidade do treinamento considerando apenas as camadas existentes, a linha vermelha contínua indica o treinamento da camada recém-inserida, e a linha azul contínua corresponde ao desempenho da rede na validação.

Observou-se que as inserções colaborativas possibilitaram a redução do erro na saída da rede neural de forma mais eficiente. Por exemplo, a adição da camada 2 mostrou-se mais vantajosa do que prolongar o treinamento da camada 1, uma vez que a nova camada conseguiu atingir erros menores em tempo significativamente menor. Esse comportamento deve-se ao fato de que cada camada recém-inserida aprende padrões residuais a partir das representações já otimizadas pelas camadas anteriores, aproveitando o conhecimento adquirido e acelerando a convergência do modelo. De forma análoga, o mesmo efeito foi observado nas demais camadas. Esses resultados evidenciam que, enquanto o treinamento prolongado das camadas antigas demandaria mais tempo para alcançar o desempenho desejado, a inserção colaborativa permite reduzir de maneira eficaz o tempo total de treinamento, mantendo ou melhorando a qualidade do modelo.

Figura 22 – Projeto de uma rede neural colaborativa.



Fonte: Elaborada pelo autor.

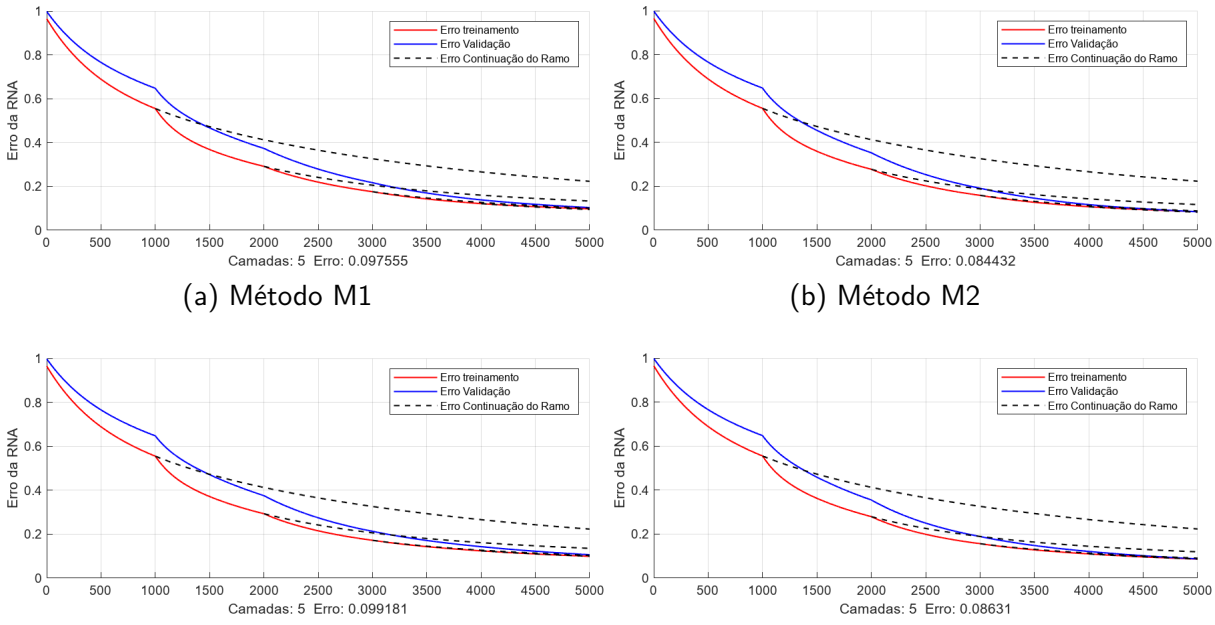
Diante do exposto, buscou-se investigar a qualidade das inserções de forma mais sistemática, analisando o impacto do grau de crescimento da rede e das conexões entre as saídas antigas e novas, conforme discutido no capítulo anterior.

Nos experimentos realizados, as arquiteturas que apresentaram melhor desempenho

possuíam nove neurônios na camada de entrada, 100 neurônios na primeira camada oculta e um neurônio na camada de saída, por se tratar de um problema de classificação binária. A taxa de aprendizado adotada foi de 1×10^{-5} entre as camadas de entrada e oculta, e de 5×10^{-5} entre a camada oculta e a camada de saída. A função de ativação empregada nas camadas ocultas foi a tangente hiperbólica, enquanto na camada de saída utilizou-se função linear. Os pesos do primeiro ramo foram inicializados com valores próximos de zero. Para o novo ramo, apenas os pesos da conexão entre a camada oculta e a camada de saída foram inicializados com zero, mantendo as demais conexões conforme a estratégia de inicialização definida para o modelo.

A Figura 23 apresenta os gráficos dos erros obtidos na inserção do tipo constante. Em todos os gráficos, observa-se que a inclusão de novas camadas resulta em uma redução significativa do erro (linha contínua vermelha) quando comparada à continuidade do treinamento das camadas antigas (linhas pretas tracejadas). Isso indica que a inserção de novas camadas não apenas acelerou o processo de busca por uma solução, mas também contribuiu para a melhoria da qualidade das representações aprendidas pelas camadas já existentes.

Figura 23 – Comparação do MSE entre os métodos com inserção constante de camadas e treinamento contínuo.



(a) Método M1 (b) Método M2

(c) Método M3 (d) Método M4

Fonte: Elaborada pelo autor.

A Tabela 1 apresenta os resultados obtidos com a inserção de camadas do tipo constante. Observa-se que o método M2 alcançou o menor valor de MSE durante o treinamento $(0,08443)$, enquanto todos os métodos apresentaram acurácia superior a 90%, destacando-se os métodos M2 e M4, que atingiram a maior precisão na classificação das amostras.

Além disso, a análise dos erros de validação e teste indica que os métodos M2 e

M4 mantêm desempenho consistente entre os conjuntos, evidenciando boa capacidade de generalização. Por outro lado, o método M1 apresenta um MSE de treino elevado, mas com erro de teste relativamente baixo, sugerindo que a rede ainda consegue generalizar, porém com menor eficiência no aprendizado inicial.

De forma geral, os resultados demonstram a robustez da abordagem de inserção constante, uma vez que diferentes métodos do mesmo tipo obtêm resultados próximos, com alta acurácia e baixa variabilidade dos erros. Entre eles, M2 destaca-se por apresentar menor MSE de treino e validação, indicando convergência mais estável, enquanto M4 atinge desempenho equivalente, mas com leve diferença nos valores de erro. Esses resultados reforçam que os métodos M2 e M4 são os mais indicados para aplicação em redes futuras, equilibrando precisão e eficiência de treinamento.

Tabela 1 – Resultados da inserção do tipo constante.

Método	**Tipo de inserção**	**Número de inserções**	**MSE Treino**	**MSE Validação**	**MSE Teste**	**Acurácia**
M1	Constante	5	0,97555	0,1033	0,08011	92,208
M2	Constante	5	**0,08443**	**0,0841**	0,0676	93,333
M3	Constante	5	0,09918	0,1060	0,0773	92,208
M4	Constante	5	0,08631	0,0874	**0,0656**	**93,333**

Fonte: Elaborada pelo autor.

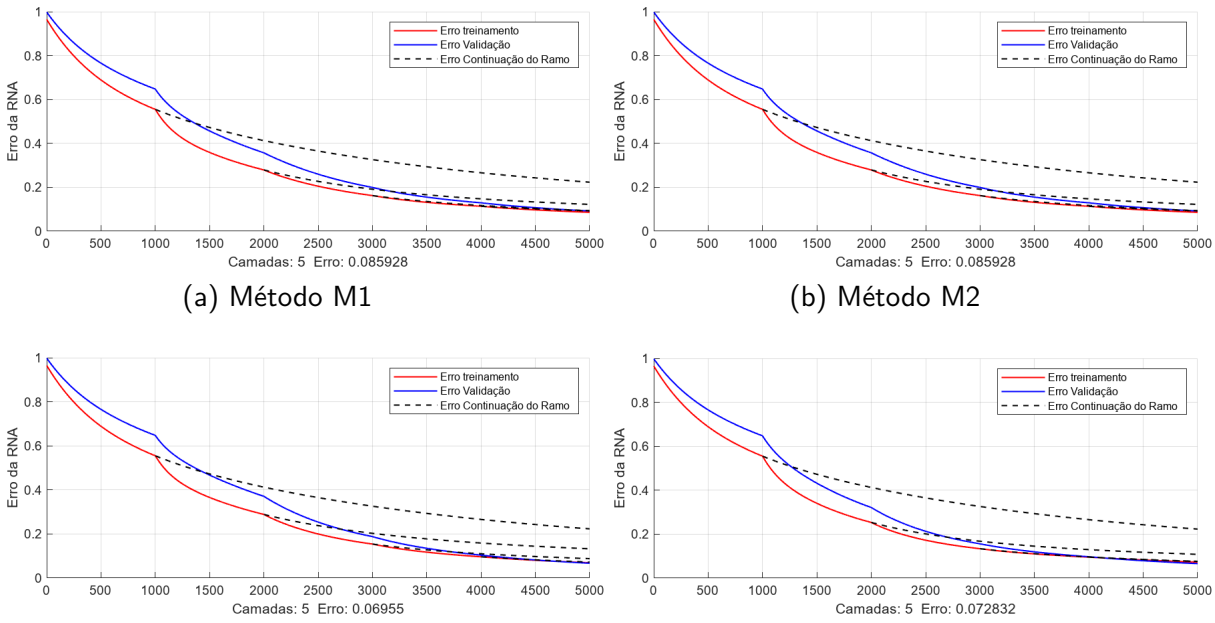
Na inserção crescente, foram obtidos os resultados ilustrados na Figura 24. Observa-se que o método M3 apresentou uma diferença considerável em relação aos outros métodos no que refere-se as inserções até a terceira camada.

A Tabela 2 apresenta os resultados obtidos com a inserção de camadas do tipo crescente. Observa-se que o método M2 alcançou a maior acurácia (97,794%) e apresentou o menor erro durante o treinamento (0,05343), indicando excelente desempenho e capacidade de generalização.

Por outro lado, o método M1 apresentou o maior erro de treinamento e a menor acurácia, evidenciando menor eficiência na adaptação à tarefa de classificação. Os métodos M3 e M4 também obtiveram resultados satisfatórios, ficando abaixo de M2, mas demonstrando que a inserção crescente, ao adicionar um ramo extra, melhora consideravelmente a performance das camadas existentes.

A análise dos erros de validação e teste confirma a consistência do desempenho, especialmente para M2 e M3, que mantêm baixa variabilidade nos diferentes conjuntos de dados. Esses resultados indicam que a abordagem de inserção crescente é robusta, capaz de acelerar o aprendizado e aprimorar a precisão das representações já existentes, sendo os métodos M2 e M3 os mais indicados para aplicação em redes futuras, equilibrando eficiência de treinamento e qualidade do modelo.

Figura 24 – Comparação do MSE entre os métodos com inserção crescente de camadas e treinamento.



(a) Método M1 (b) Método M2

(c) Método M3 (d) Método M4

Fonte: Elaborada pelo autor.

Tabela 2 – Resultados da inserção do tipo crescente.

Método	**Tipo de inserção**	**Número de inserções**	**MSE Treino**	**MSE Validação**	**MSE Teste**	**Acurácia**
M1	Crescente	5	0,08593	0.0920	0,0723	92,208
M2	Crescente	5	**0,05343**	**0,0435**	**0,0422**	**97,794**
M3	Crescente	5	0,06955	0,0674	0,0556	95,775
M4	Crescente	5	0,07283	0,0660	0,0560	93,60

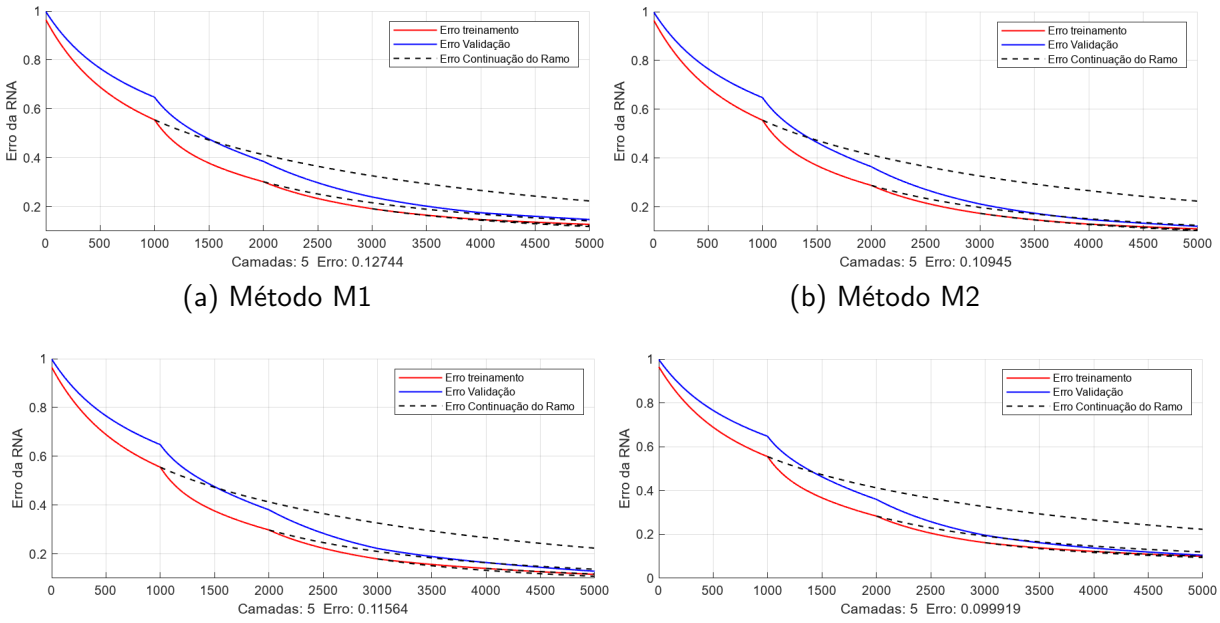
Fonte: Elaborada pelo autor.

Com a inserção decrescente, obteve-se diminuição do erro mesmo com a redução do número de neurônios. A Figura 25 demonstra que os métodos M2, M3 e M4 apresentam um queda maior que M1 com relação a inserção na segunda camada. Este fato ocorre devido a possibilidade do peso w_k ser ajustado juntamente com a existência do ramo extra que acaba compensando as diminuições da quantidade de neurônios na nova camada.

A Tabela 3 apresenta os resultados obtidos com a inserção do tipo decrescente. Nota-se que a maioria dos métodos (M1, M2 e M3) não ultrapassou a acurácia de 90%, indicando que a remoção de camadas durante o crescimento da rede compromete o desempenho geral.

O método M4 destaca-se nesse contexto, apresentando acurácia de 92,208% e menores valores de erro nos conjuntos de treino, validação e teste. Esse resultado sugere que a manutenção de um ramo extra, combinada com o ajuste adequado do peso w_k, atua como um mecanismo corretivo, mitigando os efeitos negativos da redução de camadas.

Figura 25 – Comparação do MSE entre os métodos com inserção decrescente de camadas e treinamento contínuo.



(a) Método M1 (b) Método M2

(c) Método M3 (d) Método M4

Fonte: Elaborada pelo autor.

De forma mais geral, os dados evidenciam que, embora a inserção decrescente possa reduzir a complexidade da rede, sua aplicação sem mecanismos compensatórios tende a degradar a capacidade de generalização e a precisão do modelo. Assim, para que a inserção decrescente seja efetiva, é necessário equilibrar a remoção de camadas com estratégias que preservem ou reforcem o aprendizado das representações já adquiridas.

Tabela 3 – Resultados da inserção do tipo decrescente.

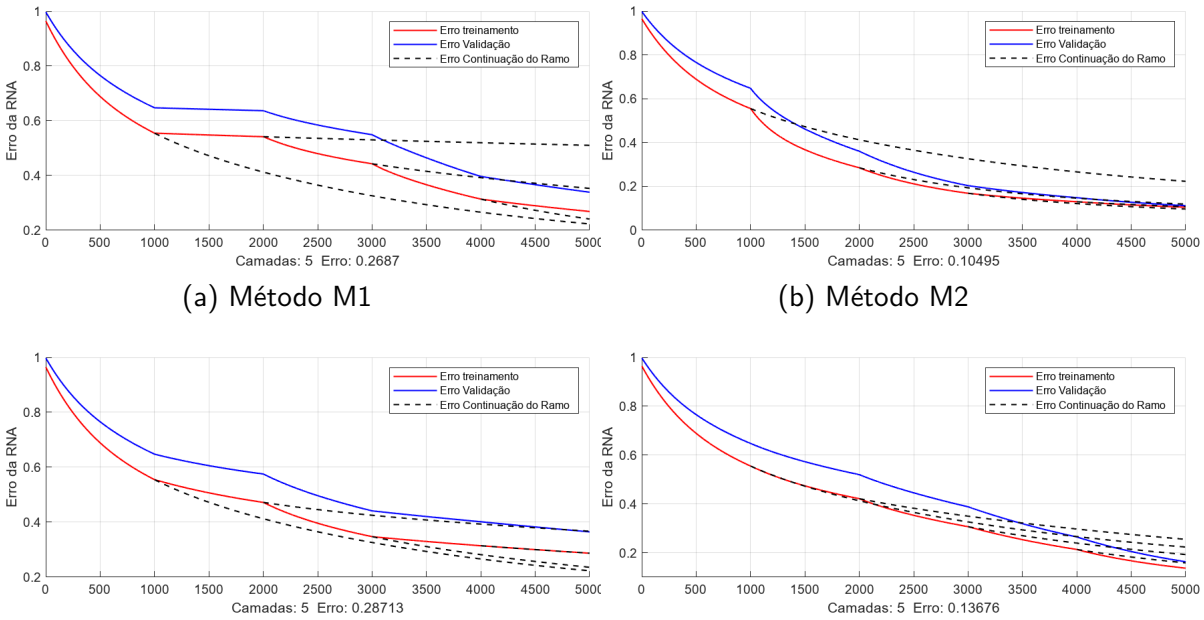
Método	**Tipo de inserção**	**Número de inserções**	**MSE Treino**	**MSE Validação**	**MSE Teste**	**Acurácia**
M1	Decrescente	5	0,12744	0,1474	0,1116	89,174
M2	Decrescente	5	0,10945	0,120	0,0933	89,174
M3	Decrescente	5	0,115638	0,1286	0,0976	89,773
M4	Decrescente	5	**0,099919**	**0,1049**	**0,0825**	**92,208**

Fonte: Elaborada pelo autor.

Na inserção aleatória, os métodos M1 e M3 não apresentaram diminuição do erro a cada nova camada inserida quando comparado á continuação do treinamento da camada um, destacando-se neste processo as camadas dois e três. Entretanto, com as camadas quatro e cinco, ocorreu uma aproximação do erro obtido na continuação camada um. Os métodos M2 e M4, com a existência do ramo externo e possibilidade do ajuste do peso w_k, conseguiram compensar as inserções aleatórias e diminuir o erro a cada nova camada inserida, com destaque

para o M4 onde as inserções foram sempre com quedas acentuadas, mesmo que o erro final não tenha sido menor que o obtido no M2.

Figura 26 – Comparação do MSE entre os métodos com inserção aleatória de camadas e treinamento contínuo.



(a) Método M1 (b) Método M2

(c) Método M3 (d) Método M4

Fonte: Elaborada pelo autor.

Por fim, a Tabela 4 mostra os resultados obtidos com a inserção de camadas do tipo aleatória. Observa-se que a variação dinâmica na quantidade de neurônios inseridos em cada etapa não promoveu melhorias consistentes no desempenho da rede. Em particular, os métodos M1 e M3 apresentaram acurácia abaixo de 80%, refletindo aumento significativo do erro e baixa capacidade de generalização.

Por outro lado, os métodos M2 e M4 demonstraram maior resiliência a esse dinamismo, mantendo acurácia acima de 88% e erros mais controlados nos conjuntos de treino, validação e teste. Esses resultados sugerem que, embora a inserção aleatória não ofereça ganhos expressivos de forma geral, o desempenho da rede pode ser significativamente influenciado pela estratégia de ajuste de pesos e pela estrutura de ramos extras presentes nos métodos mais eficientes.

Em síntese, a abordagem aleatória evidencia que a eficiência do aprendizado depende menos da quantidade de neurônios inseridos em cada etapa e mais da capacidade da rede de integrar essas alterações sem comprometer a estabilidade do modelo.

A análise dos resultados obtidos com o dataset *Breast Cancer Wisconsin* evidencia que a metodologia colaborativa é eficaz na inserção de novas camadas em redes neurais, promovendo melhora na convergência e na qualidade das representações aprendidas. Estratégias como a inserção crescente mostraram-se particularmente eficientes, permitindo à rede capturar padrões residuais de forma progressiva, enquanto a inserção constante garantiu estabilidade e

Tabela 4 – Resultados da inserção do tipo aleatória.

Método	**Tipo de inserção**	**Número de inserções**	**MSE Treino**	**MSE Validação**	**MSE Teste**	**Acurácia**
M1	Aleatório	5	0,26870	0,339185	0,2631	79,143
M2	Aleatório	5	**0,10495**	**0,111945**	**0,0928**	**91,667**
M3	Aleatório	5	0,28713	0,36486	0,2797	77,299
M4	Aleatório	5	0,13676	0,16295	0,1187	88,667

Fonte: Elaborada pelo autor.

desempenho consistente. Por outro lado, abordagens como a inserção decrescente e aleatória exigem mecanismos compensatórios, como ramos extras e ajustes de pesos, para manter a precisão e a capacidade de generalização, uma vez que a remoção ou variação dinâmica de neurônios tende a comprometer o aprendizado. De maneira geral, alguns métodos destacaram-se por maximizar o aproveitamento das camadas recém-inseridas, equilibrando eficiência de treinamento e robustez do modelo. Esses resultados demonstram que a metodologia colaborativa não apenas reduz o tempo necessário para alcançar soluções satisfatórias, como também oferece flexibilidade ao projetista da rede, que pode escolher a estratégia mais adequada sem comprometer a qualidade final do modelo.

4.2 Base de dados de áudio

Como parte da avaliação da técnica proposta, além dos experimentos realizados com a base de câncer, também foi utilizada uma base de natureza distinta, composta por sinais de fala, a fim de verificar o comportamento do método em dados não visuais e de estrutura temporal.

A base adotada foi a *Male/Female for Forced Phonetic Alignment*, disponibilizada pelo Grupo Fala Brasil. Trata-se de um conjunto formado por dois locutores (um masculino e um feminino) com 200 ocorrências de cada, gravadas em estúdio e com áudios limpos, de alta qualidade e previamente saneados (BATISTA; NETO, 2022; FALABRASIL, 2023).

O objetivo principal dessa análise é avaliar a capacidade da técnica em extrair representações discriminantes a partir de sinais acústicos, identificando como a estrutura adaptativa da rede comporta-se diante de variações fonéticas e vocais. Essa base foi escolhida por permitir observar o desempenho do modelo em um domínio de dados dinâmico e sensível a pequenas variações de padrão, complementando os resultados obtidos com dados estáticos.

Devido ao desbalanceamento existente nas classes originais, foi selecionado um subconjunto com 43 classes equilibradas, cada uma contendo 30 amostras, de modo a manter a proporcionalidade e a coerência estatística dos testes.

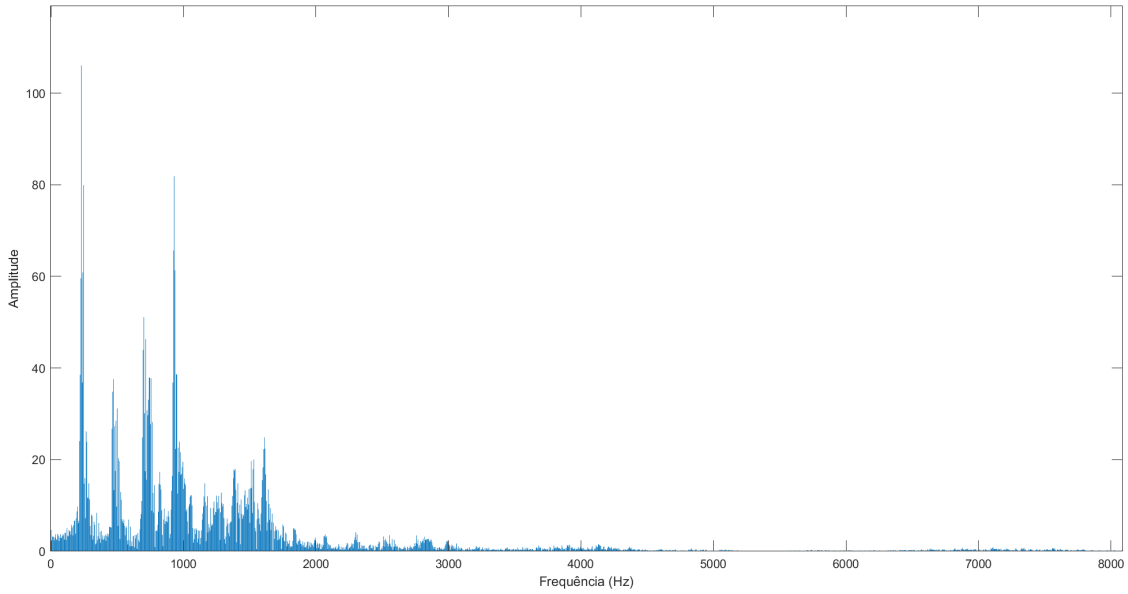
Para o pré-processamento dos áudios e preparação dos dados de entrada, foram utilizadas

as ferramentas Praat, UFPAlign e Kaldi. O Praat foi empregado para a segmentação e análise dos sinais de fala; o UFPAlign, para o alinhamento fonético automático do português brasileiro; e o Kaldi, como kit de ferramentas de reconhecimento de fala que atua em conjunto com o UFPAlign (BATISTA *et al.*, 2022; SOUZA; NETO, 2016; DIAS *et al.*, 2020).

Após o processamento, foram obtidos fragmentos correspondentes às sílabas fonéticas, os quais serviram como entradas para os experimentos realizados. Essa abordagem permitiu observar o desempenho da técnica proposta na modelagem de padrões acústicos e na adaptação dinâmica das camadas da rede neural durante o aprendizado.

Após o tratamento dos áudios realizado por Pereira (2014), aplicou-se a FFT para obter o sinal no domínio da frequência, com taxa de 16 kHz. Como o espectro gerado é simétrico e apresentado de forma duplicada no MATLAB, apenas a primeira metade contém as frequências reais, ou seja, até 8 kHz, reduzindo o custo computacional, conforme ilustrado na Figura 27.

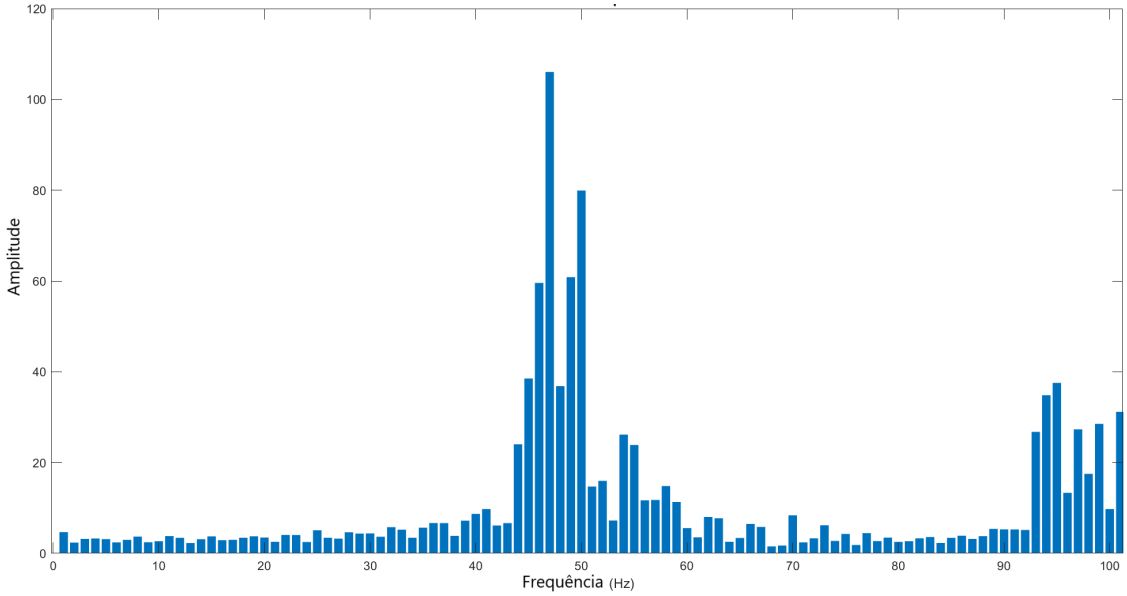
Figura 27 – FFT do sinal sonoro amostrado da sílaba fonética "da".



Fonte: Elaborada pelo autor.

Para facilitar a análise visual, os 100 primeiros dados apresentados na Figura 27 foram destacados e representados na Figura 28 em formato de barras. Cada barra foi interpretada como um objeto geométrico, permitindo uma manipulação mais estruturada dos dados. Em seguida, essas barras foram agrupadas de maneira sistemática, com o objetivo de compactar e organizar a representação das diferentes frequências presentes no sinal de cada sílaba fonética. Esse procedimento possibilita identificar padrões e variações de frequência de forma mais clara, oferecendo uma visão detalhada das características espectrais do sinal analisado. Além disso, a compactação das barras facilita comparações entre sílabas, evidenciando a distribuição energética de cada unidade fonética e contribuindo para análises subsequentes de processamento de sinais e reconhecimento de padrões acústicos.

Figura 28 – 100 primeiras frequências da sílaba fonética "da".



Fonte: Elaborada pelo autor.

No domínio da frequência, a magnitude de cada componente representa a altura da barra correspondente. A largura de cada barra foi fixada em 1, de modo que cada frequência é representada por um retângulo com largura unitária e altura proporcional à magnitude da frequência. Para simplificar e compactar a representação dessas frequências, barras adjacentes foram agrupadas, formando Figura s irregulares pela junção dos retângulos, conforme ilustrado na Figura 29.

Cada agrupamento contém o mesmo número de frequências, garantindo que todas as Figura s resultantes possuam a mesma largura. A junção dos retângulos permite determinar um único centroide para cada grupo, representado por C_m, conforme definido na Eq. 4.2. Os cálculos das coordenadas X_m e Y_m são realizados pelas Equações 4.3 e 4.4, em que a_f corresponde à área de cada retângulo, x_f é a posição da frequência (coordenada horizontal do centro do retângulo) e y_f sua magnitude (altura do retângulo). Como a base de cada retângulo é unitária, a_f equivale diretamente à magnitude da frequência.

A utilização de centroides oferece uma representação simplificada e robusta das características espectrais do sinal. Cada centroide C_m sintetiza a informação de várias frequências adjacentes em um único ponto, preservando a energia e a distribuição espectral do grupo. Essa abordagem facilita a comparação entre diferentes sílabas fonéticas, permitindo identificar padrões, picos de energia e regiões de maior relevância no espectro. Além disso, a redução do número de elementos individuais melhora a eficiência de análises subsequentes, como classificação de fonemas ou extração de características acústicas, sem perder detalhes essenciais da estrutura frequencial do sinal.

$$C_m = (X_m, Y_m) \tag{4.2}$$

$$X_m = \frac{\sum_{f=1}^{n} x_f \times a_f}{\sum_{f=1}^{n} a_f} \tag{4.3}$$

$$Y_m = \frac{\sum_{f=1}^{n} \frac{y_f}{2} \times a_f}{\sum_{f=1}^{n} a_f} \tag{4.4}$$

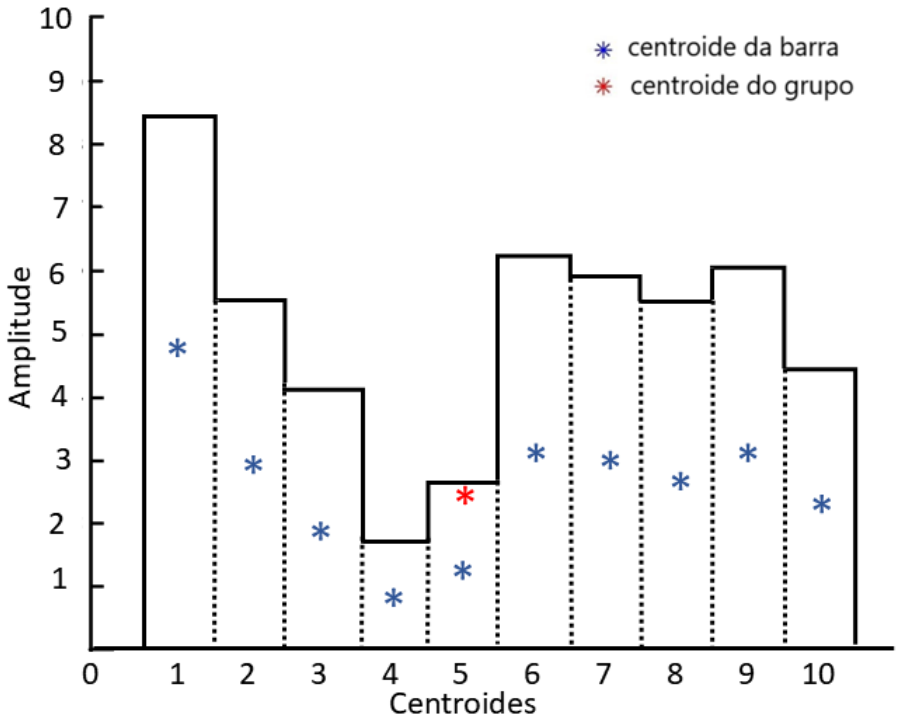
Como exemplo, a Figura 29 apresenta um gráfico formado por 10 barras, representando um único grupo de frequências. Para a construção desse gráfico, considerou-se um sinal com faixa de frequência de 8 kHz, realizando-se o agrupamento a cada 10 Hz, o que resultou em 800 grupos no total, cada um composto por 10 barras. No gráfico apresentado, os pontos azuis indicam o centro geométrico de cada barra individual, enquanto o ponto vermelho representa o centroide do grupo.

Essa operação permite compactar de forma eficiente os dados das sílabas fonéticas, reduzindo cada grupo a apenas dois valores (X_m e Y_m) e, consequentemente, diminuindo significativamente o custo computacional. Ao mesmo tempo, mantém-se a integridade das características espectrais essenciais, garantindo que a redução de dimensionalidade não comprometa de maneira crítica o aprendizado da rede neural. Assim, o uso de centroides possibilita uma representação mais concisa e organizada das frequências, preservando padrões relevantes e facilitando análises subsequentes de reconhecimento ou classificação fonética.

Para a metodologia adotada, foram considerados 16 grupos de frequências, cada um composto por 50 elementos, totalizando 16 centroides que sintetizam de forma compacta e representativa cada sílaba fonética. A escolha do agrupamento permite reduzir significativamente a quantidade de dados, preservando, entretanto, as características espectrais essenciais de cada sílaba. As coordenadas X e Y de cada centroide foram Normalizadas no intervalo de 0 a 1, tornando os dados consistentes e compatíveis com a escala de entrada exigida pela rede neural.

Essa Normalização e compactação dos dados conferem vantagens importantes: além de reduzir a dimensionalidade da entrada, diminuindo o custo computacional, elas facilitam o aprendizado da rede, pois cada centroide condensa informações sobre a distribuição de energia e a posição das frequências dentro do grupo. Com isso, é possível manter a representação das principais características acústicas das sílabas, permitindo que a rede neural capture padrões relevantes para classificação ou reconhecimento fonético sem perda significativa de informação. Além disso, o uso de centroides padronizados possibilita comparações consistentes entre diferentes sílabas e sinais, servindo como base robusta para análises subsequentes de processamento e interpretação dos dados acústicos.

Figura 29 – Centroides de um grupo de 10 frequências.



Fonte: Elaborada pelo autor.

4.2.1 Métricas usadas

Para avaliar o treinamento e o resultado da classificação, foram utilizadas as seguintes métricas: erro quadrático médio (MSE), acurácia, precisão, sensibilidade e *f1-score*.

O MSE foi utilizado para otimizar os parâmetros $W_1^i, b_1^i, W_2^i, b_2^i$ de uma função custo θ_f durante o treinamento. Ele permitiu acompanhar a capacidade da rede a cada nova camada inserida, avaliando o nível de ajuste dos pesos e a contribuição de cada camada na redução do erro total da rede. Já a acurácia forneceu uma medida geral da proporção de classificações corretas realizadas pelo modelo.

A precisão (PRE), ou valor preditivo positivo, pode ser definida como sendo a relação de acertos verdadeiros do modelo em relação às classificações realizadas. A PRE é dada pela Eq. 4.5.

$$PRE = \frac{VP}{VP + FP} \tag{4.5}$$

A sensibilidade (SEN) ou *recall*, considera a proporção de todos os positivos reais que foram classificados corretamente como positivos. Portanto, a SEN é dada pela Eq. 4.6.

$$SEN = \frac{VP}{VP + FN} \tag{4.6}$$

O *f1-score* considera tanto o cálculo da precisão quanto o da sensibilidade, sendo composto pela média harmônica de ambas as métricas. Ele é definido pela Eq. 4.7.

$$f1score = 2 * \frac{PRE * SEN}{PRE + SEN} \tag{4.7}$$

Enquanto o MSE forneceu uma medida do ajuste interno da rede durante o treinamento, precisão, sensibilidade e *f1-score* possibilitaram avaliar a qualidade final do aprendizado da rede para o problema proposto, considerando tanto o equilíbrio entre classes quanto a capacidade de reconhecimento correto de cada categoria.

4.2.2 Avaliação dos experimentos

Durante os experimentos, foram avaliadas diferentes estratégias de crescimento da rede colaborativa *stacked autoencoder*, variando a quantidade de neurônios inseridos na nova camada oculta em relação à camada anterior. As estratégias adotadas incluíram crescimento constante, crescente, decrescente e aleatório. Além da configuração do número de neurônios, também foi investigado o ajuste do peso k, responsável pela conexão entre a camada oculta do novo ramo e a camada de saída do ramo previamente existente, permitindo analisar o impacto dessa parametrização no desempenho da rede.

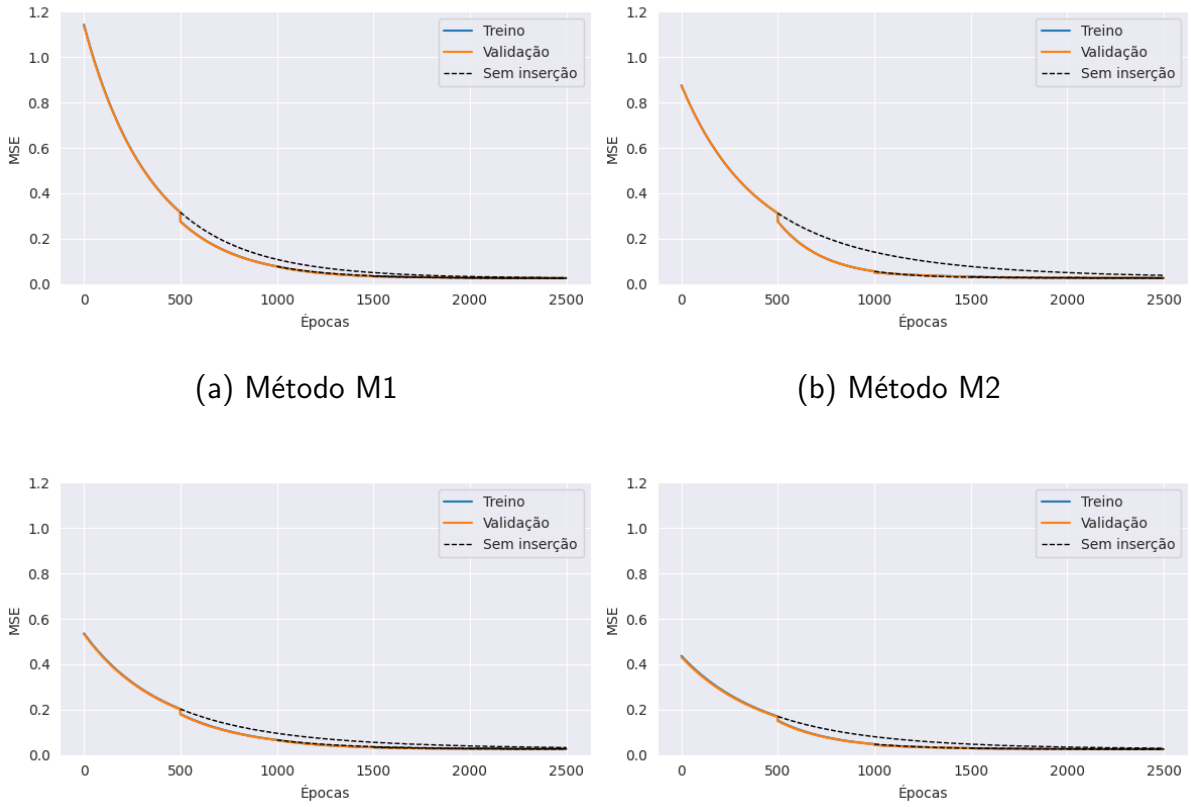
Para cada método de inserção, foram adicionadas quatro camadas ocultas de forma sequencial, e o desempenho da rede foi avaliado no problema de classificação proposto. Ao final do processo de crescimento, todas as redes consistiam em uma camada de entrada, cinco camadas ocultas e uma camada de saída, conFigura ndo a arquitetura completa utilizada nos experimentos.

As redes treinadas receberam como entradas as coordenadas (X_m, Y_m) dos 16 centroides calculados para cada sílaba fonética, totalizando 32 entradas. A primeira camada oculta inicializou-se com 25 neurônios, enquanto a camada de saída possuía 43 neurônios, adequados à natureza de múltiplas classes do problema. As taxas de aprendizado foram definidas como 1×10^{-5} entre a camada de entrada e a primeira camada oculta, e 5×10^{-5} entre a camada oculta final e a camada de saída.

Cada ramo da rede utilizou a função de ativação tangente hiperbólica nas camadas ocultas e função linear na camada de saída, garantindo não linearidade suficiente para aprendizado e preservação da escala dos valores preditos. A inicialização dos pesos da primeira camada oculta ocorreu aleatoriamente em torno do zero, enquanto, para os novos ramos adicionados, apenas os pesos da conexão da camada oculta com a camada de saída foram inicializados com zero, mantendo as demais conexões inalteradas. Essa estratégia permitiu isolar o efeito do novo ramo no ajuste da rede, facilitando a análise do impacto da adição de neurônios na melhoria do desempenho.

A Figura 30 apresenta os gráficos dos erros obtidos durante o treinamento na inserção de camadas do tipo constante em rede *stacked autoencoder*. Nesses gráficos, as linhas azuis representam o erro de saída (treinamento) da rede usado para ajustes dos pesos e, paralelamente, as linhas laranjas representam o erro de validação. Destaca-se que as linhas azuis permanecem com valore próximos e abaixo das linhas laranjas ao longo do treinamento, indicando que o erro nos dados de treinamento é inferior ao erro nos dados de validação, como esperado. Enquanto que as linhas pretas tracejadas simbolizam o erro de saída da rede para ajustes dos pesos desconsiderando a inserção de uma nova camada. Isto é, a linha tracejada representa a continuação do treinamento de um ramo sem considerar inserção de uma nova camada. Portanto, foi possível analisar o ganho obtido com a inserção de uma nova camada.

Figura 30 – Gráficos do MSE para inserção do tipo constante e diferentes métodos.



(a) Método M1

(b) Método M2

(c) Método M3

(d) Método M4

Fonte: Elaborada pelo autor.

A Tabela 5 apresenta os resultados da inserção do tipo constante de cinco camadas ocultas. Com relação ao erro, durante o treinamento, os métodos M1 e M4 apresentaram os menores valores (0,026), porém no teste os métodos M2 e M4 apresentaram os menores valores. Em todo caso, os métodos ficaram com valores muitos próximos.

Na inserção do tipo crescente, foram obtidos os resultados ilustrados na Figura 31. Os métodos M2 e M4 apresentaram os menores erros em relação aos outros métodos. A inserção

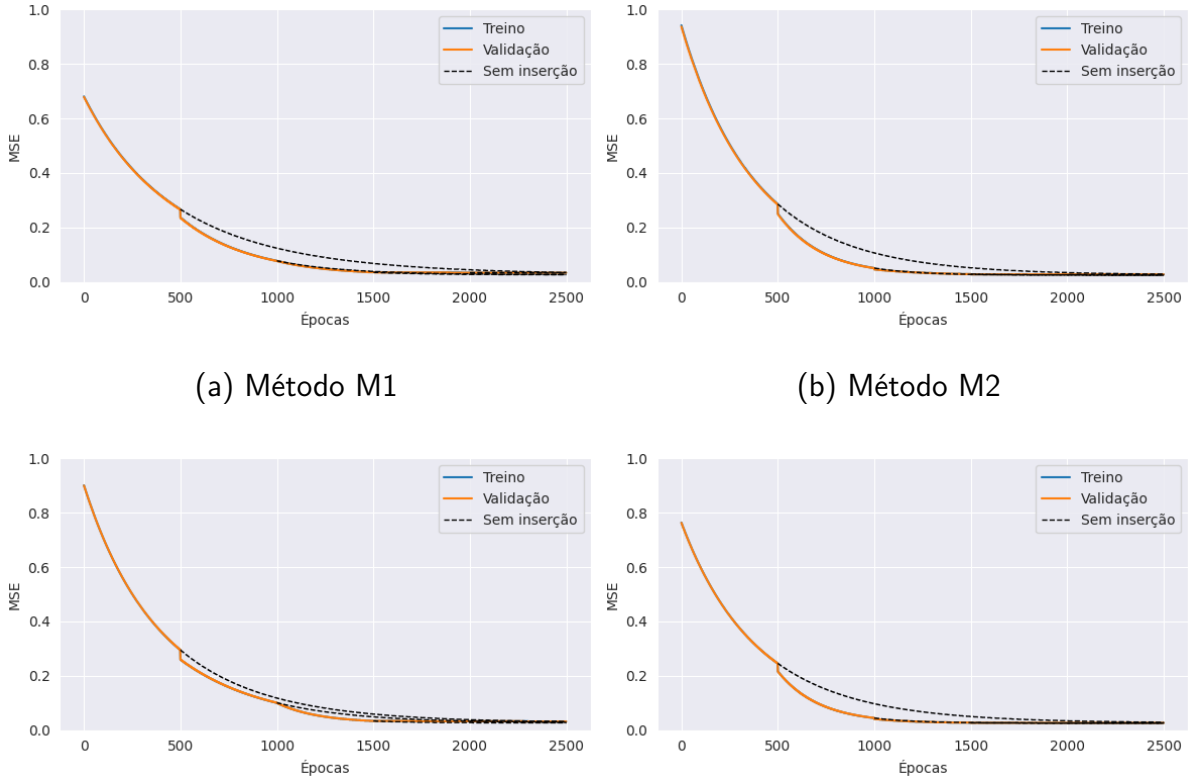
Tabela 5 – Inserção do tipo constante de acordo com a quantidade de camadas.

Método	Uma camada			Duas camadas			Três camadas			Quatro camadas			Cinco camadas		
	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste
M1	0,317	0,315	0,320	0,078	0,077	0,079	0,034	0,034	0,035	0,028	0,029	**0,026**	0,026	0,026	0,027
M2	0,312	0,312	0,307	0,056	0,056	0,055	0,033	0,032	0,031	0,028	0,028	0,027	0,027	0,026	0,026
M3	0,203	0,201	0,203	0,067	0,066	0,067	0,034	0,034	0,034	0,028	0,028	0,028	0,027	0,027	0,027
M4	**0,170**	**0,168**	**0,171**	**0,048**	**0,047**	**0,049**	**0,031**	**0,030**	**0,031**	**0,027**	**0,027**	0,027	**0,026**	**0,206**	**0,026**

Fonte: Elaborada pelo autor.

do tipo crescente apresentou no método M1 e M3 maiores erros durante o seu treinamento (0,034 e 0,032, respectivamente). Com isso, observou-se que permitir o ajuste do peso k oferece melhorias no treinamento.

Figura 31 – Gráficos do MSE para inserção do tipo crescente e diferentes métodos.



(a) Método M1

(b) Método M2

(c) Método M3

(d) Método M4

Fonte: Elaborada pelo autor.

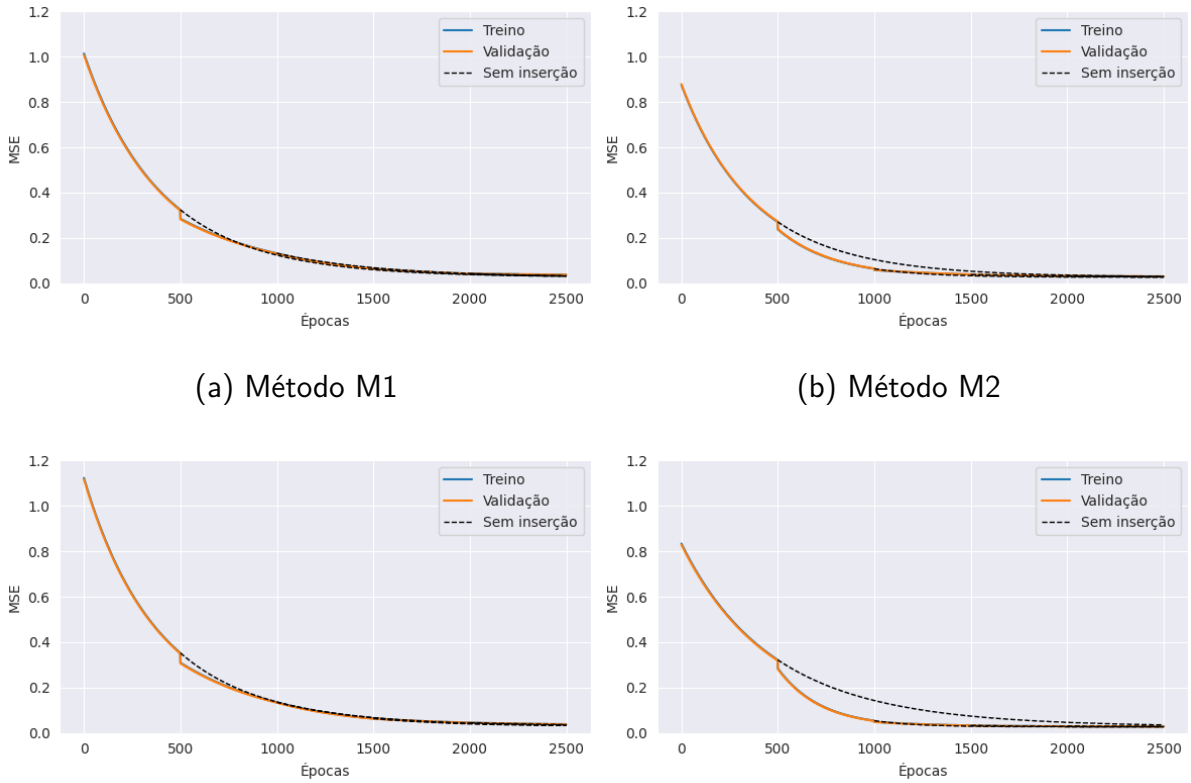
Com a inserção decrescente, obteve-se diminuição do erro no treinamento mesmo com a redução do número de neurônios. A Figura 32 demonstra que os métodos M2 e M4 apresentaram um queda maior que nos métodos M1 e M3. Na Tabela 7, tem-se que M1 e M3 com erro no treinamento de 0,037 e 0,038 após as cincos camadas inseridas. Este fato ocorreu devido a possibilidade dos pesos w_k ser ajustado e a existência do ramo externo que tende a compensar as diminuições da quantidade de neurônios nas novas camadas.

Tabela 6 – Inserção do tipo crescente de acordo com a quantidade de camadas.

Método	Uma camada			Duas camadas			Três camadas			Quatro camadas			Cinco camadas		
	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste
M1	0,267	0,266	0,266	0,078	0,077	0,077	0,036	0,036	0,036	0,034	0,034	0,034	0,034	0,034	0,034
M2	0,286	0,284	0,289	0,051	0,051	0,052	0,029	0,029	0,029	0,028	0,028	0,028	0,028	0,028	0,028
M3	0,295	0,296	0,294	0,101	0,101	0,100	0,034	0,034	0,033	0,033	0,033	0,032	0,032	0,032	0,032
M4	**0,246**	**0,246**	**0,245**	**0,045**	**0,045**	**0,044**	**0,027**	**0,028**	**0,027**	**0,027**	**0,027**	**0,027**	**0,027**	**0,027**	**0,027**

Fonte: Elaborada pelo autor.

Figura 32 – Gráficos do MSE para inserção do tipo decrescente e diferentes métodos.



(a) Método M1

(b) Método M2

(c) Método M3

(d) Método M4

Fonte: Elaborada pelo autor.

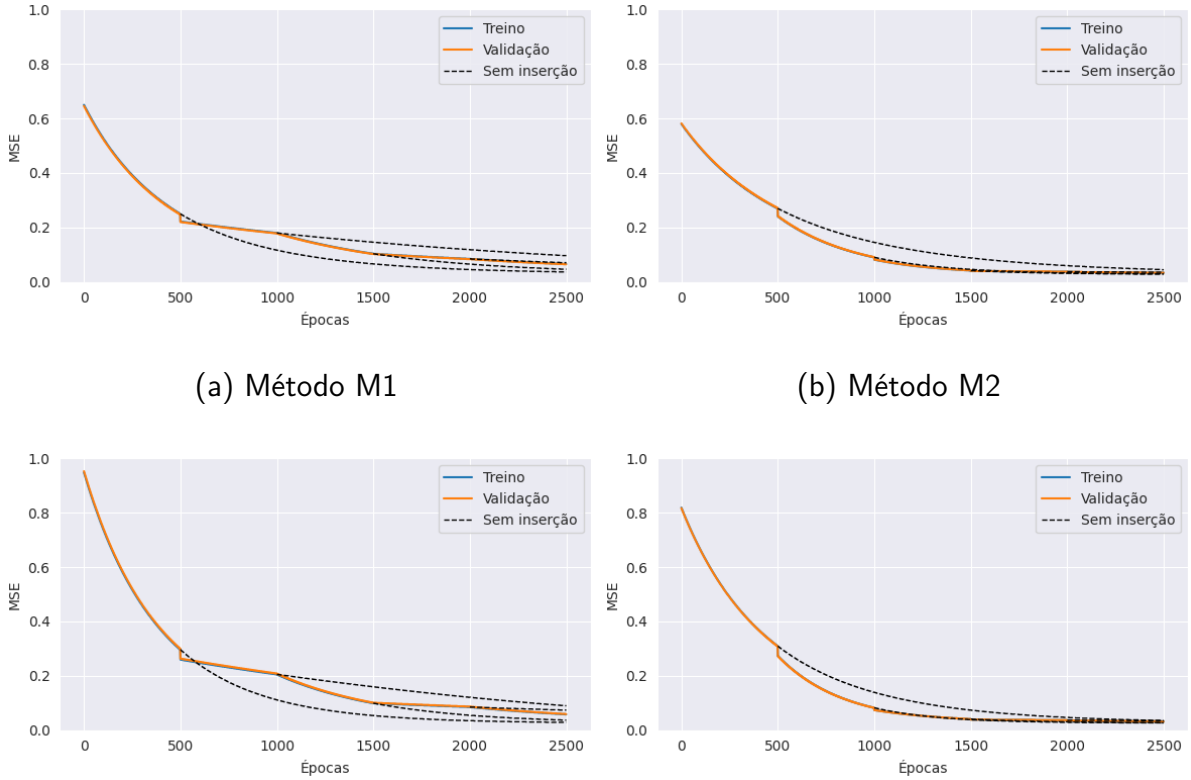
Tabela 7 – Inserção do tipo decrescente de acordo com a quantidade de camadas.

Método	Uma camada			Duas camadas			Três camadas			Quatro camadas			Cinco camadas		
	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste
M1	0,324	0,321	0,325	0,132	0,130	0,132	0,065	0,064	0,065	0,042	0,042	0,042	0,037	0,037	0,037
M2	**0,270**	**0,272**	**0,266**	0,064	0,065	0,062	0,038	0,039	0,037	0,032	0,033	0,031	0,030	0,030	0,029
M3	0,352	0,350	0,346	0,135	0,134	0,131	0,064	0,063	0,061	0,046	0,045	0,043	0,038	0,038	0,037
M4	0,322	0,319	0,321	**0,054**	**0,053**	**0,054**	**0,033**	**0,033**	**0,033**	**0,030**	**0,029**	**0,029**	**0,029**	**0,028**	**0,028**

Fonte: Elaborada pelo autor.

Na inserção do tipo aleatória ilustrado na Figura 33, observa-se que os métodos M1 e M3 não apresentaram diminuição do erro de treinamento a cada nova camada inserida, quando comparados à continuação do treinamento da primeira camada. Porém, com a camada três, ocorreu uma diminuição do erro obtido na continuação da primeira camada. Os métodos M2 e M4, com a existência do ramo externo e possibilidade do ajuste do peso k, conseguiram compensar as inserções aleatórias e diminuir o erro a cada nova camada inserida, com destaque para o método M4 onde as inserções foram sempre com quedas acentuadas.

Figura 33 – Gráficos do MSE para inserção do tipo aleatória e diferentes métodos.



(a) Método M1

(b) Método M2

(c) Método M3

(d) Método M4

Fonte: Elaborada pelo autor.

Na Tabela 8, têm-se os dados de inserção aleatória. Observou-se que o dinamismo entre a quantidade de neurônios em cada inserção não oferece melhorias significativas. Não obstante, os métodos M2 e M4, cujos valores do erro no treinamento foram 0,034 e 0,032, tendem a compensar fortemente este dinamismo, quando comparados aos métodos M1 e M3 que apresentaram erro no treinamento de 0,066 e 0,058.

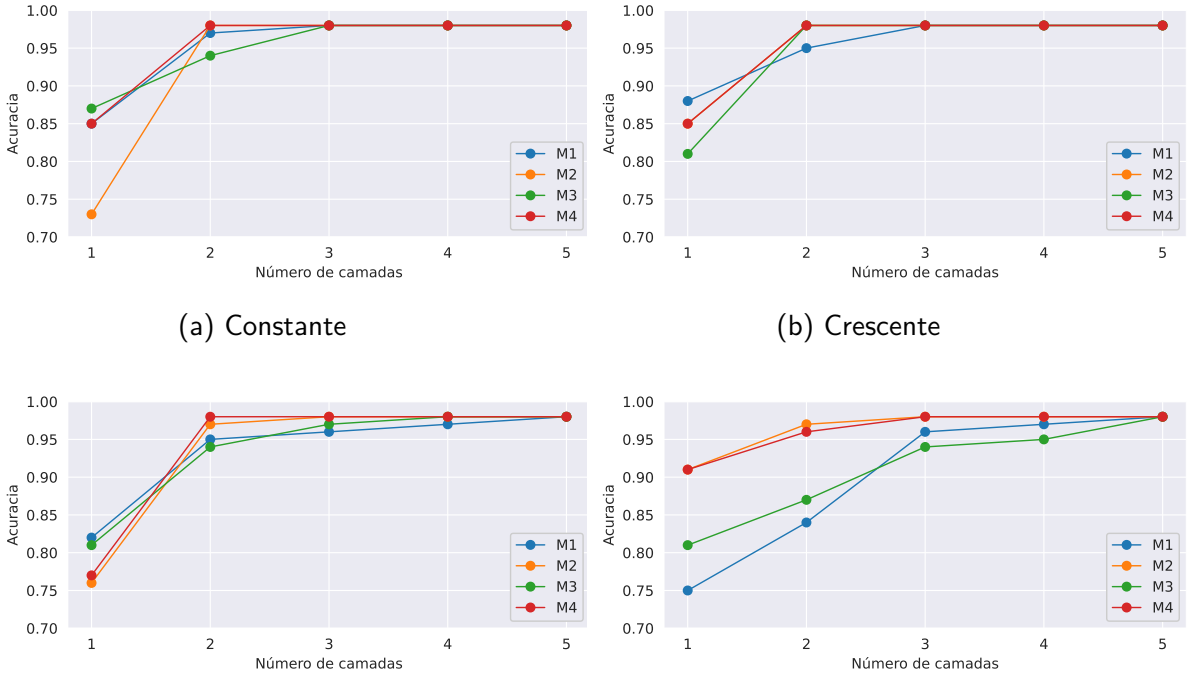
A Figura 34 apresenta os resultados da acurácia de acordo com o tipo de inserção. Nas Tabelas 9, 10, 11 e 12, têm-se os valores obtidos da acurácia para cada inserção. As inserções do tipo constante, crescente e decrescente conseguiram com três camada inseridas apresentar acurácia de 98%, enquanto que a inserção do tipo aleatória apresenta acurácia abaixo desse valor, somente conseguindo atingir na maioria dos métodos após a inserção de quatro camadas.

Tabela 8 – Inserção do tipo aleatória de acordo com a quantidade de camadas.

Método	Uma camada			Duas camadas			Três camadas			Quatro camadas			Cinco camadas		
	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste	MSE treino	MSE valid.	MSE teste
M1	**0,250**	**0,248**	**0,249**	0,180	0,178	0,179	0,101	0,103	0,103	0,085	0,084	0,084	0,066	0,065	0,065
M2	0,270	0,271	0,269	0,090	0,091	0,090	0,042	0,043	0,042	0,038	0,038	0,038	0,034	0,034	0,034
M3	0,296	0,300	0,298	0,205	0,208	0,207	0,099	0,101	0,101	0,085	0,087	0,087	0,058	0,059	0,059
M4	0,309	0,308	0,310	**0,082**	**0,082**	**0,083**	**0,041**	**0,041**	**0,041**	**0,037**	**0,037**	**0,037**	**0,032**	**0,032**	**0,032**

Fonte: Elaborada pelo autor.

Figura 34 – Gráficos das acurácias de acordo com o tipo de inserção.



(a) Constante

(b) Crescente

(c) Decrescente

(d) Aleatória

Fonte: Elaborada pelo autor.

Tabela 9 – Acurácia para inserção do tipo constante.

Método	Camada 1	Camada 2	Camada 3	Camada 4	Camada 5
M1	0,79	0,94	0,98	0,98	0,98
M2	0,77	**0,98**	0,98	0,98	0,98
M3	0,84	0,95	0,98	0,98	0,98
M4	**0,93**	0,97	**0,98**	**0,98**	**0,98**

Fonte: Elaborada pelo autor.

Tabela 10 – Acurácia para inserção do tipo crescente.

Método	Camada 1	Camada 2	Camada 3	Camada 4	Camada 5
M1	0,83	0,93	0,98	0,98	0,98
M2	0,84	0,98	0,98	0,98	0,98
M3	**0,85**	0,97	0,98	0,98	0,98
M4	0,82	**0,98**	**0,98**	**0,98**	**0,98**

Fonte: Elaborada pelo autor.

Tabela 11 – Acurácia para inserção do tipo decrescente.

Método	Camada 1	Camada 2	Camada 3	Camada 4	Camada 5
M1	0,86	0,93	0,97	0,98	0,98
M2	**0,87**	0,95	0,98	0,98	0,98
M3	0,80	0,94	0,98	0,98	0,98
M4	0,73	**0,97**	**0,98**	**0,98**	**0,98**

Fonte: Elaborada pelo autor.

Tabela 12 – Acurácia para inserção do tipo aleatória.

Método	Camada 1	Camada 2	Camada 3	Camada 4	Camada 5
M1	**0,87**	0,87	0,94	0,97	0,97
M2	0,85	0,94	0,98	0,98	0,98
M3	0,83	0,87	0,93	0,94	0,97
M4	0,76	**0,97**	**0,98**	**0,98**	**0,98**

Fonte: Elaborada pelo autor.

Com relação aos trabalhos da literatura, a Tabela 13 apresenta a comparação entre o modelo proposto (SAE CollabNet) e a versão anterior (Collabnet), conforme apresentado em (PEREIRA, 2014). As métricas de desempenho acurácia, sensibilidade, precisão e *f1-score* foram utilizadas para avaliar o comportamento dos classificadores, considerando o melhor resultado obtido em cada experimento.

Observa-se que o modelo SAE CollabNet apresentou ganhos expressivos em todas as métricas quando comparado à versão anterior. A acurácia evoluiu de 0,76 para 0,98, indicando uma melhoria significativa na taxa de acertos globais. A sensibilidade e a precisão também apresentaram aumentos consistentes, evidenciando que o novo método consegue identificar corretamente as amostras positivas e reduzir os falsos positivos de forma equilibrada. O *f1-score* reforça a estabilidade do desempenho, demonstrando bom equilíbrio entre precisão e revocação.

Outro ponto relevante é que o modelo proposto mantém alto desempenho independentemente do tipo de inserção realizado durante o processo construtivo. Nos testes com inserções aleatórias, os métodos M1 e M3 atingiram valores mínimos de acurácia de 87%, superando com folga o melhor resultado da versão anterior. Esse comportamento mostra que a nova abordagem colaborativa é mais robusta frente às variações de estrutura e consegue preservar a capacidade de generalização mesmo em cenários menos previsíveis.

Em síntese, a SAE CollabNet demonstra avanços claros em estabilidade, precisão e capacidade de adaptação, consolidando-se como uma evolução consistente da arquitetura original e validando a eficiência do mecanismo colaborativo de inserção de camadas.

4.3 Avaliação de mutação da função de ativação

Nas versões tradicionais de redes autoencoder, a camada de saída costuma empregar uma função de ativação linear. Essa escolha é adequada quando o objetivo é reconstruir sinais

Tabela 13 – Comparação com outros trabalhos da literatura.

Métrica	Collabnet	SAE Collabnet
acurácia	0,76	0,98
sensibilidade	0,76	0,98
precisão	0,82	0,93
f1-score	0,79	0,91

Fonte: Elaborada pelo autor.

contínuos e não normalizados, pois evita distorções no espaço de saída e facilita a convergência do erro de reconstrução. No entanto, essa configuração também impõe uma limitação: a rede tende a restringir sua capacidade de representação, especialmente quando as camadas internas utilizam funções não lineares. A presença de uma camada final linear pode reduzir o potencial de modelagem das relações complexas aprendidas nas camadas anteriores, funcionando como um gargalo que suaviza as respostas do modelo.

Para investigar esse possível gargalo e verificar se a função linear realmente restringe o poder expressivo do modelo, foi avaliado o efeito de uma **mutação da função de ativação** da camada de saída. A mutação consiste em substituir a ativação linear por uma função não linear, como *sigmoid*, *tanh* ou *ReLU*, de forma controlada, mantendo fixos os demais parâmetros da arquitetura. Essa modificação altera a natureza da transformação de saída, permitindo observar se o ganho de não linearidade contribui para uma melhor adaptação da rede ao processo de aprendizado colaborativo e à inserção de novas camadas.

O procedimento foi aplicado dentro do contexto construtivo da rede, em que novas camadas ocultas são adicionadas incrementalmente. Assim, a mutação foi analisada em dois momentos distintos:

- ChangeOut: a mutação ocorre enquanto a camada ainda é a camada de saída, ou seja, antes da inserção da próxima camada oculta. Nessa condição, avalia-se o impacto da introdução da não linearidade à capacidade de reconstrução e a estabilidade do treinamento.
- ChangeHidden: a mutação é aplicada após a camada deixar de ser a camada de saída, tornando-se uma camada oculta intermediária. Essa análise permite observar se a mudança da ativação, quando feita tardiamente, influencia a propagação de informação e a transferência de representações entre camadas.
- Normal: durante o treinamento não ocorre a mutação de função de ativação. Assim, a função da camada de saída permanece do tipo linear.

Essa abordagem permite compreender se o momento da mutação (antes ou depois da

conversão da camada) interfere na adaptabilidade da rede e no equilíbrio entre linearidade, estabilidade e poder de generalização. Em síntese, busca-se avaliar se a limitação imposta por uma função de saída linear é relevante em arquiteturas que evoluem de forma construtiva e colaborativa.

4.3.1 Conjunto de dados e critérios de seleção

Os experimentos foram realizados utilizando bases de dados obtidas do conjunto de benchmarking *OpenML-CC18* (BISCHL *et al.*, 2019), que reúne coleções padronizadas para a avaliação de algoritmos de aprendizado de máquina. A partir desse conjunto, foram selecionados quatro bases de dados de acordo com critérios específicos, a fim de garantir a consistência experimental: cada base continha entre 500 e 1000 instâncias, de 20 a 25 atributos, e exatamente duas classes-alvo. Essas restrições foram definidas para equilibrar o custo computacional e a representatividade dos dados, permitindo que os modelos fossem treinados em condições comparáveis, mas com complexidade suficiente para avaliar a capacidade de generalização da rede.

As redes começaram com uma camada de entrada, uma primeira camada oculta com 100 neurônios e uma camada de saída com um único neurônio. Uma taxa de aprendizado de 10^{-5} foi usada tanto para as conexões de entrada para a camada oculta quanto para as conexões da camada oculta para a camada de saída. As camadas ocultas empregaram a função de ativação tangente hiperbólica (tanh); a camada de saída usou uma ativação linear. Cada procedimento de inserção de um novo ramo foi executado por até 500 épocas.

4.3.2 Análise dos resultados

A Figura 35 apresenta a acurácia média à medida que novas camadas são adicionadas, considerando os tipos de métodos utilizados nos diferentes conjuntos de dados e a presença ou ausência de mutação na função de ativação da camada de saída do ramo.

O método ChangeHidden alcança a maior acurácia, atingindo aproximadamente 0,765 nas camadas quatro e cinco. ChangeOut apresenta desempenho ligeiramente inferior ao ChangeHidden, mas supera a configuração Normal a partir da segunda camada. A configuração Normal estabiliza em torno de 0,755 a partir da terceira camada.

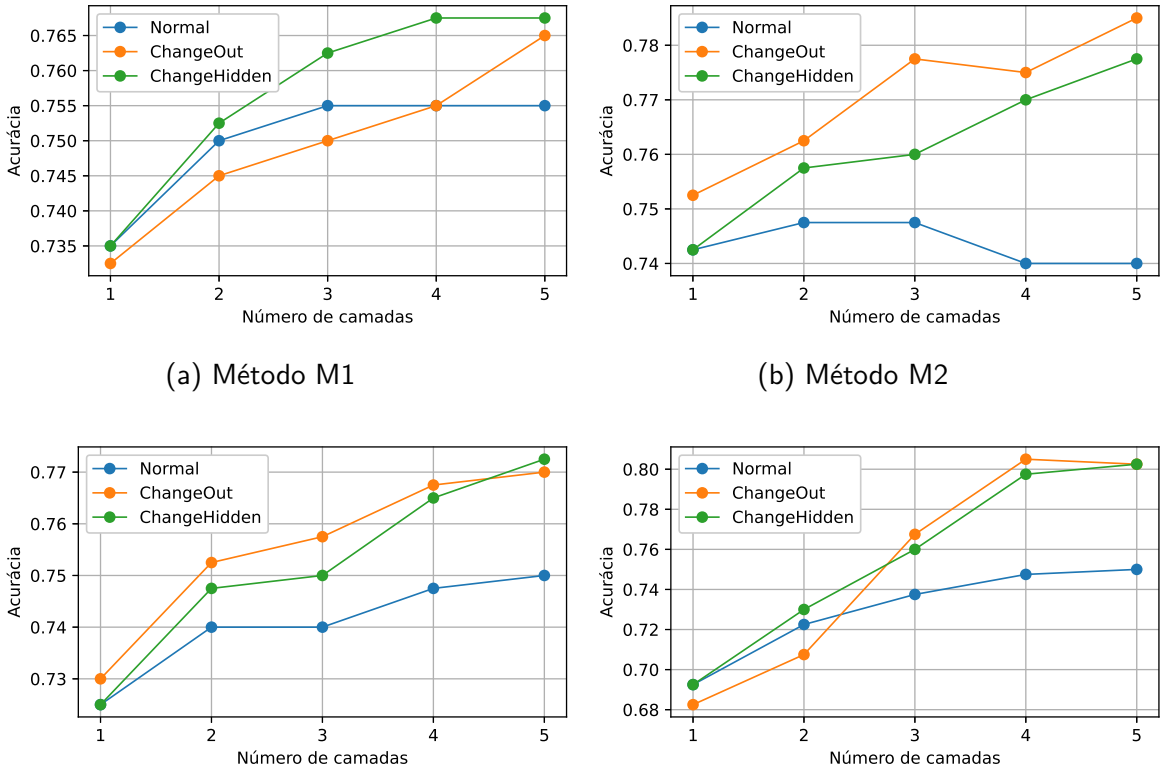
No método M2, ChangeOut atinge a acurácia máxima, cerca de 0,785 na quinta camada. ChangeHidden também apresenta melhora consistente, embora ligeiramente abaixo do ChangeOut. Já a configuração Normal saturou precocemente em torno de 0,745 a partir da segunda camada.

Para o método M3, todas as configurações apresentam ganhos de acurácia com o aumento de camadas. ChangeOut e ChangeHidden apresentam desempenho semelhante e superam a configuração Normal, que cresce de forma modesta e se estabiliza próxima de 0,75.

No método M4, ChangeOut e ChangeHidden aumentam rapidamente a acurácia, alcançando cerca de 0,80 nas camadas quatro e cinco. Em contraste, a estratégia Normal cresce mais lentamente e se estabiliza abaixo de 0,75.

De forma geral, ChangeOut e ChangeHidden superam consistentemente a estratégia Normal à medida que a profundidade da rede aumenta. Os ganhos mais expressivos ocorrem entre as camadas um e três, indicando que as estratégias modificadas são particularmente eficazes nas primeiras fases de aprofundamento da rede. Entre elas, ChangeHidden tende a ser competitiva ou superior ao ChangeOut, especialmente em arquiteturas mais profundas. Já a estratégia Normal apresenta a menor acurácia em todas as conFigura ções e responde minimamente ao aumento da profundidade, evidenciando sua limitada escalabilidade em modelos mais profundos.

Figura 35 – Acurácia média em função do número de camadas inseridas.



(a) Método M1

(b) Método M2

(c) Método M3

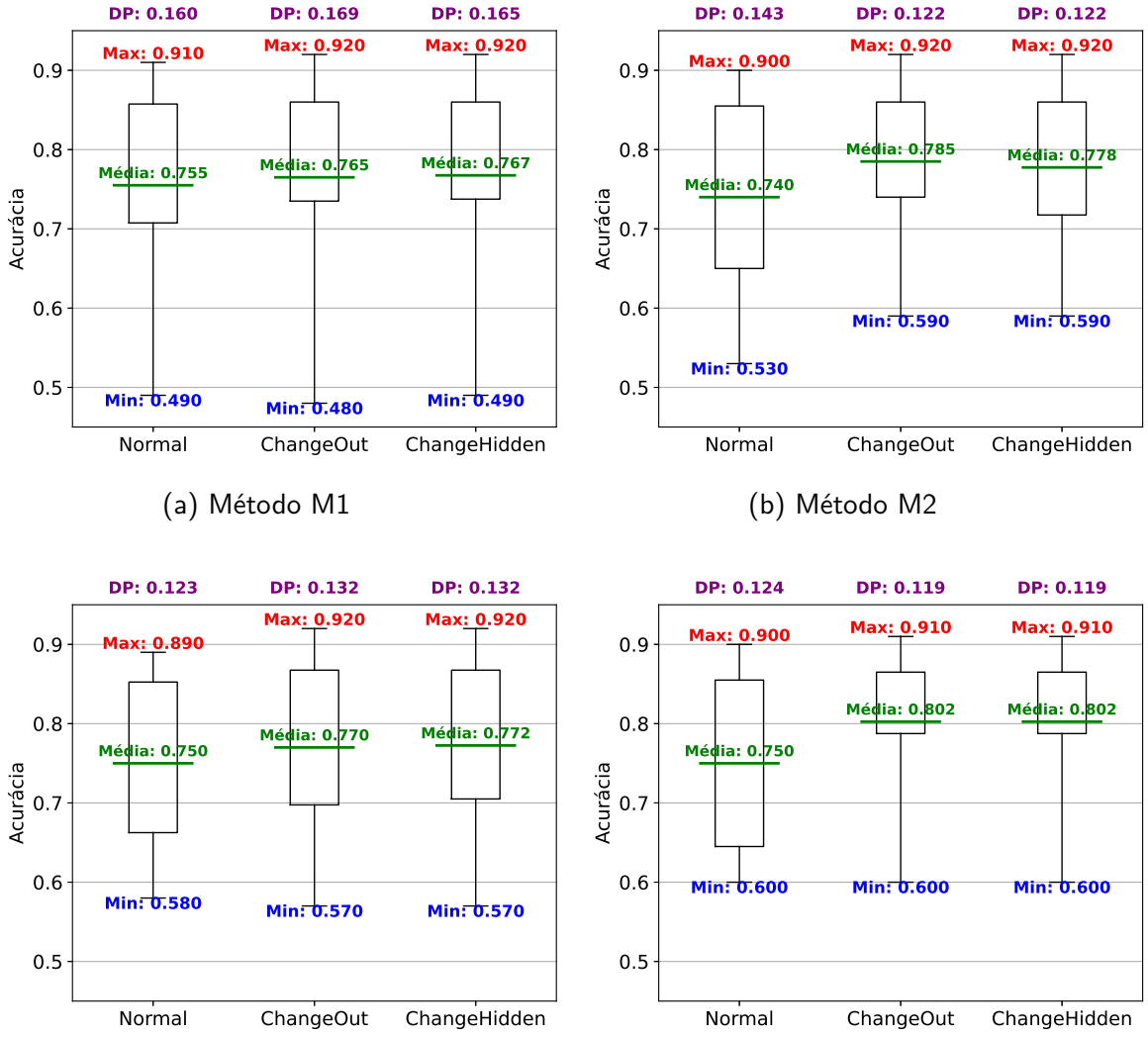
(d) Método M4

Fonte: Elaborada pelo autor.

Os gráficos da Figura 36 comparam a acurácia de teste em três cenários distintos de função de ativação da camada de saída: Normal, ChangeOut e ChangeHidden. Cada figura apresenta os valores de média, desvio padrão, mínimo e máximo de acurácia obtidos em cada condição, permitindo observar a variação de desempenho entre os diferentes tipos de mutação aplicados.

Na Figura 36a, a acurácia média é semelhante para os cenários Normal e ChangeOut

Figura 36 – Análise da acurácia em função do tipo de mutação da função de ativação.



(a) Método M1

(b) Método M2

(c) Método M3

(d) Método M4

Fonte: Elaborada pelo autor.

(ambos com 0,775), enquanto o cenário ChangeHidden apresenta uma média ligeiramente inferior (0,747). A variabilidade, medida pelo desvio padrão, é maior em ChangeHidden (0,185), seguida de ChangeOut (0,169) e Normal (0,160). Isso indica que, apesar da estabilidade na média, os cenários modificados, especialmente o ChangeHidden, apresentam maior dispersão nos resultados.

Na Figura 36b, tanto ChangeOut quanto ChangeHidden superam o cenário Normal em acurácia média (0,765 e 0,766 contra 0,740) e exibem menor variabilidade (desvios padrão de 0,122 e 0,143, respectivamente), indicando melhor desempenho e maior consistência.

Na Figura 36c, a acurácia média aumenta para ambas as modificações (0,770 para ChangeOut e ChangeHidden em comparação a 0,750 para Normal), porém o cenário ChangeOut apresenta a maior variabilidade (desvio padrão de 0,157), seguido de ChangeHidden (0,132) e

Normal (0,123).

Por fim, a Figura 36d mostra as maiores acurácias médias para ChangeOut e ChangeHidden (ambos com 0,802), com menores desvios padrão (0,119) em relação ao Normal (0,124). Assim, observa-se que os cenários ChangeOut e ChangeHidden proporcionam melhorias significativas na acurácia e na estabilidade das predições, sendo que o ChangeHidden apresenta vantagem em relação à estratégia Normal.

4.4 *Framework* Experimental com a *Benchmark Suite* OpenML-CC18

A base de dados OpenML-CC18 é amplamente utilizada para avaliação de algoritmos de aprendizado de máquina. Ela reúne tarefas de classificação de diferentes domínios, tamanhos, números de atributos e níveis de desbalanceamento, seguindo uma estrutura padronizada para permitir avaliações reprodutíveis, utilizando protocolos experimentais bem definidos, como validação cruzada estratificada, métricas padronizadas e separação clara entre dados de treinamento e teste (BISCHL *et al.*, 2019).

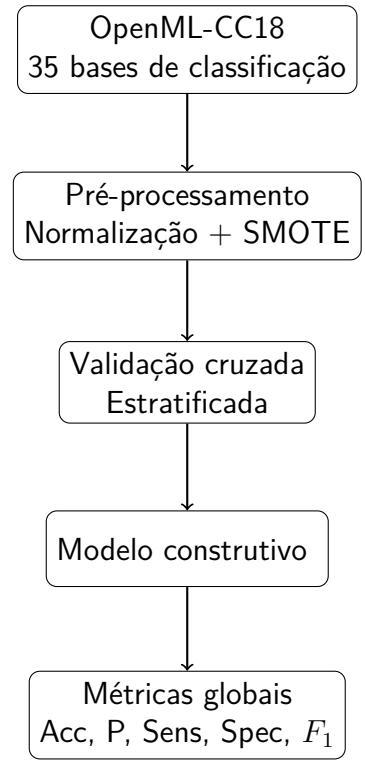
Neste trabalho, a base CC18 foi empregada para avaliar a arquitetura SAE CollabNet em 35 tarefas de classificação binária. Os dados foram normalizados no intervalo [0,1]. As bases com desbalanceamento acentuado passaram pela técnica SMOTE para correção do desbalanceamento, aplicado apenas quando necessário para evitar impacto negativo no treinamento, especialmente em arquiteturas incrementais. Todo o processo utilizou validação cruzada estratificada de 5 *kfolds*, garantindo maior robustez estatística, conforme resumido no *pipeline* da Figura 37. Após essa etapa, o modelo é treinado e avaliado usando métricas típica de problemas de classificação.

4.4.1 Integração e Pré-processamento Automático dos Conjuntos de Dados

A integração com a base *CC18* foi realizada automaticamente via API, garantindo acesso padronizado aos dados e metadados. Para isso, o pré-processamento seguiu o seguinte procedimento:

- Download e verificação: obtenção via API e checagem de atributos duplicados, colunas constantes e inconsistências estruturais.
- Tratamento de valores ausentes: imputação por média (atributos numéricos) e moda (atributos categóricos).
- Codificação e Normalização: *one-hot encoding* para atributos categóricos e Normalização para atributos numéricos.

Figura 37 – *Pipeline* experimental utilizando o benchmark OpenML-CC18 para avaliação do modelo construtivo.



Fonte: Elaborada pelo autor.

- Balanceamento: aplicação de *SMOTE* somente quando havia forte desbalanceamento entre classes.

Todo o *pipeline* foi implementado em *Python*, garantindo reprodutibilidade e permitindo focar na avaliação da arquitetura SAE CollabNet.

A utilização da base *OpenML-CC18* em problemas de classificação binária permitiu avaliar o comportamento da rede SAE CollabNet em um conjunto diversificado de domínios, com diferentes níveis de complexidade, tamanhos e distribuições de classes. Essa abordagem forneceu uma visão ampla sobre a robustez e a estabilidade dos métodos de inserção implementados no modelo.

4.4.2 Métricas e Protocolo Experimental

A SAE CollabNet foi configurada com cinco estágios, cada um composto por um bloco principal de 50 neurônios e um bloco auxiliar fixo de 20 neurônios (ramo extra). O tamanho do bloco principal variou conforme o tipo de inserção definido: inserção constante, inserção crescente, inserção decrescente ou inserção dinâmica. Cada estratégia alterou apenas a quantidade de neurônios adicionada em cada estágio, mantendo o restante da arquitetura

inalterado. Todos os treinamentos dos ramos ocorreram até 500 épocas, totalizando 2500 épocas a criação da rede.

As funções de ativações seguiram o padrão *tanh–linear*, com uso de viés no primeiro e terceiro blocos. O treinamento empregou gradiente descendente estocástico com taxas de aprendizado de 1×10^{-5} e 5×10^{-5} na camada oculta e camada de saída, aplicando treinamento por ciclo a cada camada inserida e posterior ajuste conjunto. Os hiperparâmetros foram mantidos fixos em todas as execuções.

A avaliação utilizou validação cruzada estratificada com $k = 5$. As métricas geradas em cada iteração foram armazenadas, e ao final calcularam-se média e desvio-padrão.

As métricas utilizadas foram:

- Acurácia: proporção de acertos.
- Precisão: confiança nas classificações positivas.
- Sensibilidade: capacidade de identificar instâncias positivas.
- Especificidade: capacidade de identificar instâncias negativas.
- *F1-score:* média harmônica entre precisão e sensibilidade.

Com relação aos recursos computacionais e ferramentas, os experimentos foram realizados no Google *Colab*, utilizando GPU NVIDIA Tesla T4 e Python 3.10, garantindo execução eficiente e reprodutível.

4.4.3 Análise Geral dos Resultados

As Figuras 38 e 39, analisadas em conjunto, reforçam o impacto existente no tipo de inserção no desempenho e na estabilidade da acurácia média nos conjuntos de *datasets analisados*. A Figura 38 evidencia a evolução da acurácia ao longo das cinco camadas, enquanto a Figura 39 apresenta a dispersão, os valores médios, máximos, mínimos e a variabilidade dos resultados para cada método.

No tipo constante (CNT), observa-se crescimento consistente da acurácia com o aumento das camadas, acompanhado por médias mais elevadas e menor dispersão. O método M4 apresenta não apenas as maiores médias(0,64), mas também máximos elevados e variância controlada, indicando desempenho superior e estável. O método M3 mantém médias inferiores (aproximadamente 0,51), confirmando menor eficiência.

Na inserção do tipo crescente (CRT), os ganhos progressivos observados na Figura 38 são corroborados pela Figura 39, que mostra médias intermediárias e dispersão moderada. Novamente, o M4 destaca-se com melhores médias e máximos consistentes (0,54 e 0,93),

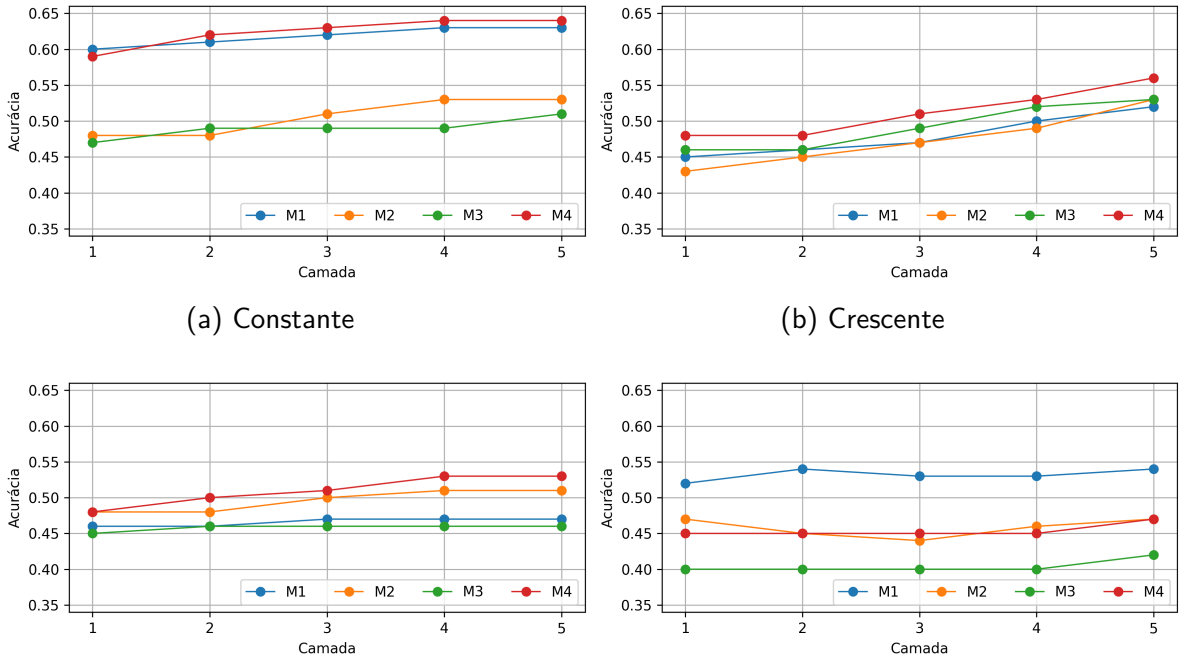
enquanto os demais métodos apresentam desempenho próximo entre si, porém inferior ao cenário constante.

No tipo decrescente (DRT), observa-se a saturação precoce da acurácia, o que é confirmado pela existência de médias mais baixas e maior variabilidade, especialmente nos métodos M1, M2 e M3. O M4 mantém vantagem relativa, mas com aumento do desvio padrão, evidenciando menor estabilidade do aprendizado nesse tipo de inserção.

Por fim, na inserção do tipo dinâmica/aleatória (ALT), a ausência de uma tendência clara de crescimento na Figura 38 reflete-se na Figura 39 por maiores dispersões e amplitudes entre valores mínimos e máximos. Nesse cenário, o M1 apresenta maior robustez média (0,54), enquanto os demais métodos sofrem maior degradação e instabilidade.

Em síntese, as duas figuras mostram de forma complementar que estratégias de inserção estruturadas, especialmente a constante e a crescente, produzem não apenas maiores valores médios de acurácia, mas também resultados mais estáveis. O método M4 é o mais consistente na maioria dos cenários, enquanto inserções decrescentes e dinâmicas comprometem tanto o desempenho quanto a confiabilidade do aprendizado. Este fato evidência que conexão de ramo extra e ajuste de dos pesos de conexão entre as camadas de saída de cada ramo contribuem efetivamente para o aprendizado.

Figura 38 – Desempenho da acurácia de acordo com o tipo de crescimento da rede.



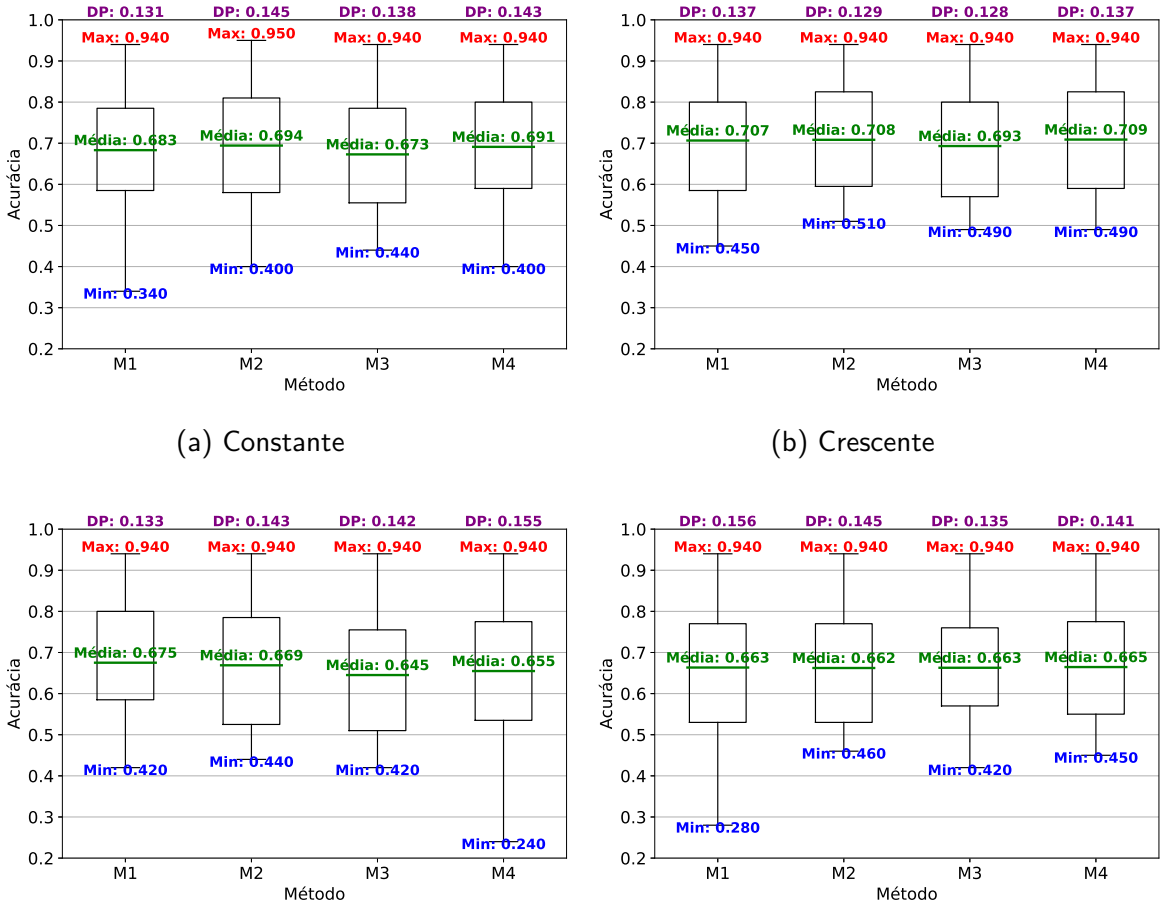
(a) Constante (b) Crescente

(c) Decrescente (d) Aleatória

Fonte: Elaborada pelo autor.

Na Figura 40, tem-se o desempenho da acurácia em função do número de camadas para cada método, considerando os diferentes tipos de inserção. De modo geral, observa-se que todos

Figura 39 – Distribuição da acurácia de acordo com o tipo de inserção.



(a) Constante

(b) Crescente

(c) Decrescente

(d) Aleatória

Fonte: Elaborada pelo autor.

os métodos beneficiam-se do aumento do número de camadas, porém com comportamentos distintos quanto à taxa de crescimento e estabilidade.

No método M1, há crescimento praticamente linear da acurácia para todos os tipos de inserção. O tipo CRT apresenta o maior ganho absoluto, alcançando aproximadamente 0,707 na quinta camada, enquanto CNT e DRT mostram evolução mais moderada, com ambos permanecendo próximo de 0,68. O tipo ALT mantém desempenho inferior, embora também apresente tendência de crescimento,com valor final em torno de 0,663.

No método M2, o crescimento é mais acentuado nas primeiras camadas, especialmente para CRT e CNT, indicando maior capacidade de adaptação inicial. O tipo ALT parte de valores significativamente menores, aproximadamente 0,694, mas apresenta recuperação progressiva, reduzindo a diferença nas camadas finais, alcançando 0,662. Ainda assim, CRT permanece como o tipo mais eficiente, atingindo cerca de 0,708 na quinta camada.

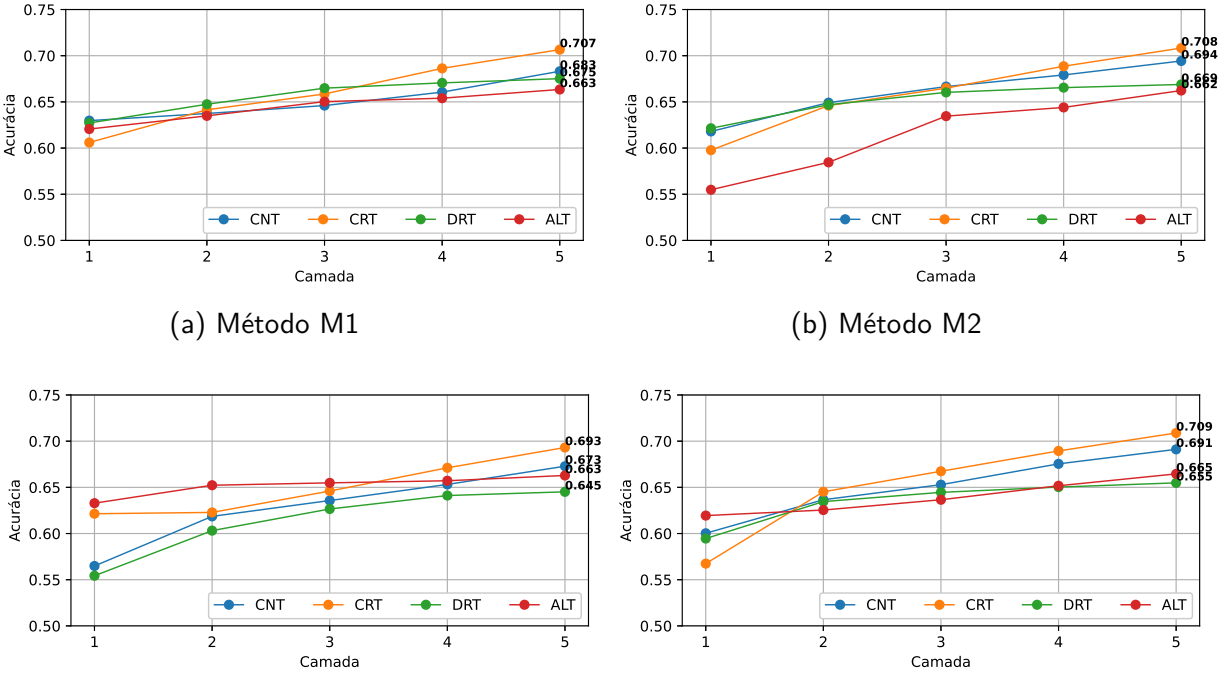
No método M3, observa-se maior sensibilidade ao tipo de inserção, o que culmina em um crescimento mais moderado. O desempenho inicial é inferior aos métodos anteriores,

principalmente para CNT e DRT, porém há ganhos consistentes com o aumento das camadas. O tipo CRT novamente destaca-se, com valor proximo de 0,693, enquanto ALT apresenta crescimento mais lento e menor acurácia final, permanecendo em torno de 0,655.

No método M4, verifica-se o melhor equilíbrio entre crescimento e estabilidade. Todos os tipos de inserção apresentam evolução consistente, com destaque para CRT, que atinge a maior acurácia final entre todos os cenários analisados. Os tipos CNT e DRT mantêm desempenho competitivo, enquanto ALT apresenta crescimento mais contido, porém estável.

De modo geral, a Figura 40 evidencia que o tipo de inserção CRT é o mais eficaz de forma consistente em todos os métodos, especialmente quando combinado com o método M4, que apresenta os melhores resultados globais. Além disso, o aumento do número de camadas contribui positivamente para o desempenho, desde que a combinação de método e o tipo de inserção favoreçam um crescimento estruturado do modelo.

Figura 40 – Desempenho da acurácia de acordo com o método.



(a) Método M1

(b) Método M2

(c) Método M3

(d) Método M4

Fonte: Elaborada pelo autor.

Na Figura 41, apresentam-se as métricas globais do modelo para os diferentes tipos de inserção, permitindo uma avaliação conjunta do desempenho médio, da variabilidade e dos valores extremos obtidos em cada abordagem.

Na especificidade e na sensibilidade, observa-se comportamento praticamente idêntico entre os tipos de inserção. As médias situam-se em torno de 0,52 a 0,53, com desvios padrão próximos de 0,09 a 0,10, indicando desempenho equilibrado, porém moderado. Os valores

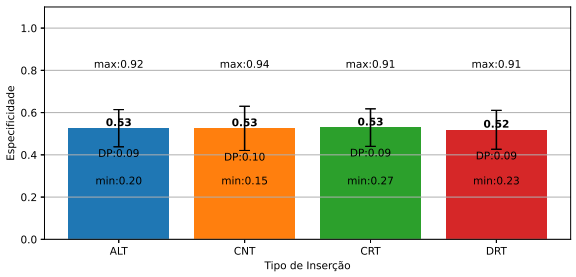
máximos atingem até 0,94 no tipo CNT e 0,92 no tipo ALT, enquanto os valores mínimos variam entre 0,15 e 0,27, evidenciando variabilidade entre execuções.

Na métrica de precisão, o tipo CRT apresenta o melhor desempenho médio, com valor aproximado de 0,54, além de alcançar o maior valor máximo observado (0,96). Os demais tipos de inserção (ALT, CNT e DRT) apresentam médias próximas, entre 0,52 e 0,53, com desvios padrão ligeiramente superiores, indicando maior dispersão dos resultados.

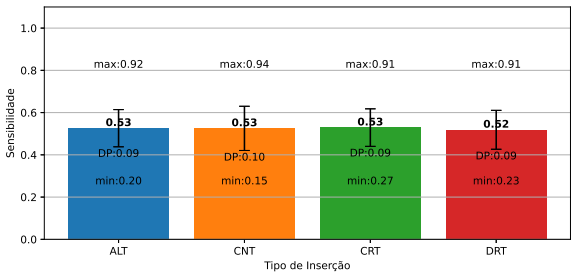
Quanto ao F1-score, os tipos CNT e CRT destacam-se com médias em torno de 0,51, superando ALT e DRT, que permanecem próximos de 0,50. Os desvios padrão mantêm-se relativamente baixos, entre 0,10 e 0,12, sugerindo consistência nos resultados, embora os valores mínimos indiquem degradação pontual em alguns cenários.

De forma geral, a Figura 41 mostra que, apesar das diferenças médias entre os tipos de inserção serem reduzidas, o tipo CRT apresenta vantagem consistente nas métricas combinadas, especialmente na precisão e no F1-score, além de alcançar os maiores valores máximos, indicando melhor equilíbrio entre desempenho e robustez global do modelo.

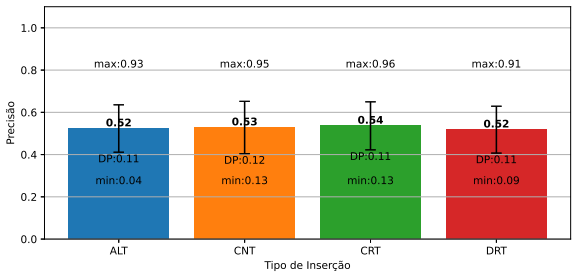
Figura 41 – Métricas globais do modelo.



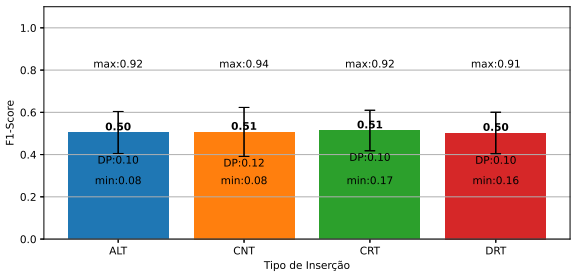
(a) Especificidade



(b) Sensibilidade



(c) Precisão



(d) F1-score

Fonte: Elaborada pelo autor.

A Figura 42 analisa o comportamento da acurácia considerando simultaneamente o tamanho do dataset e o tipo de inserção, permitindo avaliar como a disponibilidade de dados influencia o desempenho e a estabilidade do modelo sob diferentes estratégias de crescimento. A utilização de gráficos de violino possibilita observar não apenas valores centrais, mas também a dispersão, a assimetria e a concentração das distribuições de acurácia em cada cenário, fornecendo uma visão mais completa do processo de aprendizado. Essa análise é fundamental

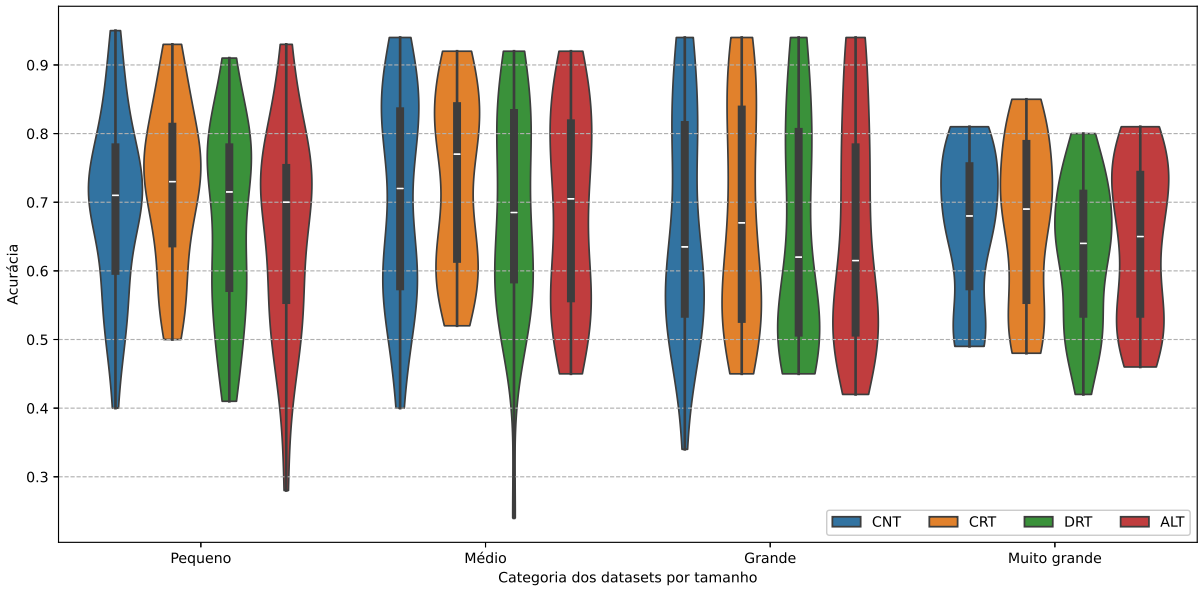
para identificar quais tipos de inserção são mais robustos frente a variações no volume de dados e como o aumento do dataset contribui para a consolidação do desempenho do modelo.

Analisando a Figura 42, oberva-se que o tamanho do dataset impacta fortemente a distribuição da acurácia, independentemente do tipo de inserção. Para datasets pequenos e médios, observa-se maior dispersão das acurácias, com caudas longas e valores mínimos mais baixos, indicando instabilidade do aprendizado. Nesse cenário, o tipo CRT tende a apresentar medianas ligeiramente superiores e distribuições mais concentradas, enquanto ALT e DRT exibem maior variabilidade.

Em datasets grandes, há melhora geral das medianas, mas ainda com dispersão relevante, sobretudo para CNT e ALT. O CRT mantém desempenho consistente, com maior concentração em faixas altas de acurácia, sugerindo melhor aproveitamento do volume de dados.

Nos datasets muito grandes, as distribuições tornam-se mais estreitas para todos os tipos de inserção, indicando ganho de estabilidade. Ainda assim, o CRT preserva as maiores medianas e quartis superiores, enquanto DRT apresenta desempenho inferior e maior concentração em valores médios.

Figura 42 – Desempenho da acurácia de acordo com o tamanho das bases de dados e o tipo de inserção.



Fonte: Elaborada pelo autor.

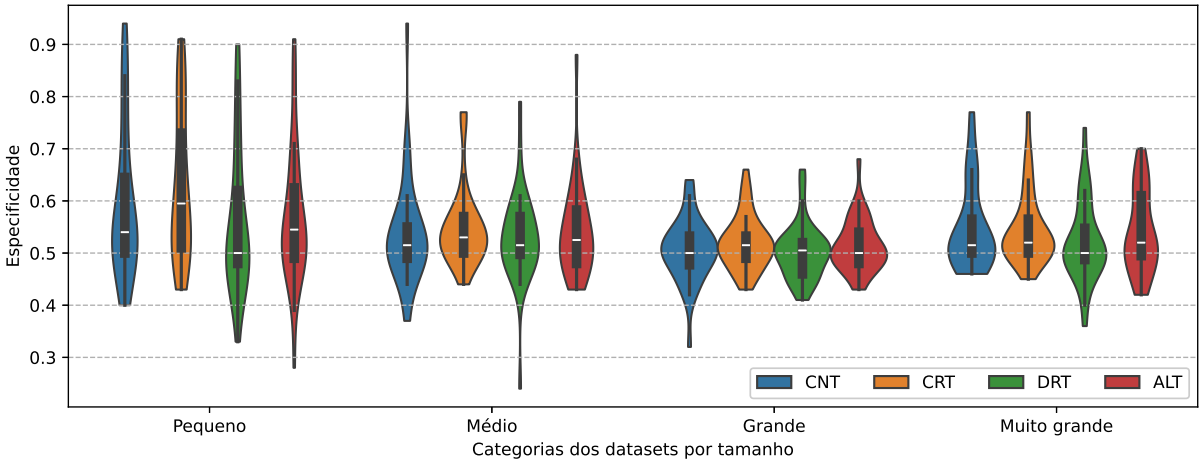
As Figuras 43, 44, 45 e 46 apresentam as métricas globais de especificidade, sensibilidade, precisão e F_1-score em função do tamanho do conjunto de dados utilizado nas etapas de treinamento e teste. De forma geral, observa-se um impacto direto do tamanho do dataset sobre o desempenho do modelo. À medida que o conjunto de dados evolui de Pequeno para Grande e Muito grande, as métricas tendem a apresentar valores médios ligeiramente superiores e menor

variabilidade, indicando maior estabilidade e robustez do aprendizado. Esse comportamento é consistente em todas as métricas analisadas, sugerindo que o aumento da quantidade de dados contribui para a redução de flutuações no desempenho.

Ao comparar as métricas, nota-se que a especificidade e a precisão apresentam, em média, os melhores resultados, com valores centrais concentrados entre 0,6 e 0,7 para a maioria dos tamanhos de dataset. Em contraste, a sensibilidade e o F_1-score exibem valores médios mais baixos, geralmente situados entre 0,5 e 0,6, indicando maior dificuldade do modelo em manter simultaneamente altas taxas de detecção e equilíbrio entre precisão e sensibilidade.

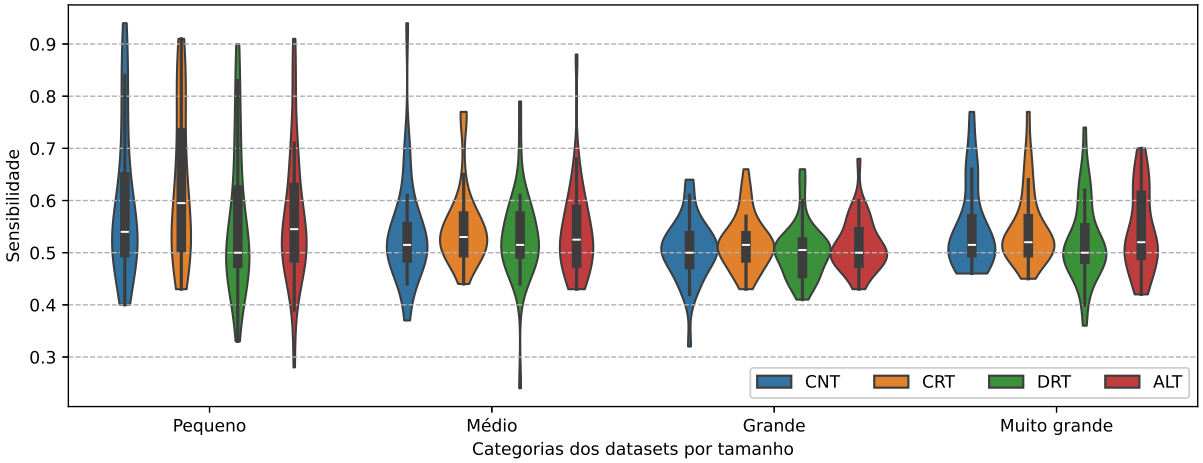
Diferenças relevantes também são observadas entre as categorias de dados. Para datasets pequenos, as categorias CNT e ALT tendem a apresentar medianas mais elevadas nas métricas avaliadas, enquanto CRT e DRT exibem desempenho inferior. Com o aumento do tamanho do dataset, essas diferenças tornam-se menos acentuadas, embora ainda persistam variações entre as categorias, evidenciando que o desempenho do modelo depende não apenas do volume de dados, mas também das características específicas de cada categoria.

Figura 43 – Especificade do modelo de acordo com o tamanho das bases de dados.



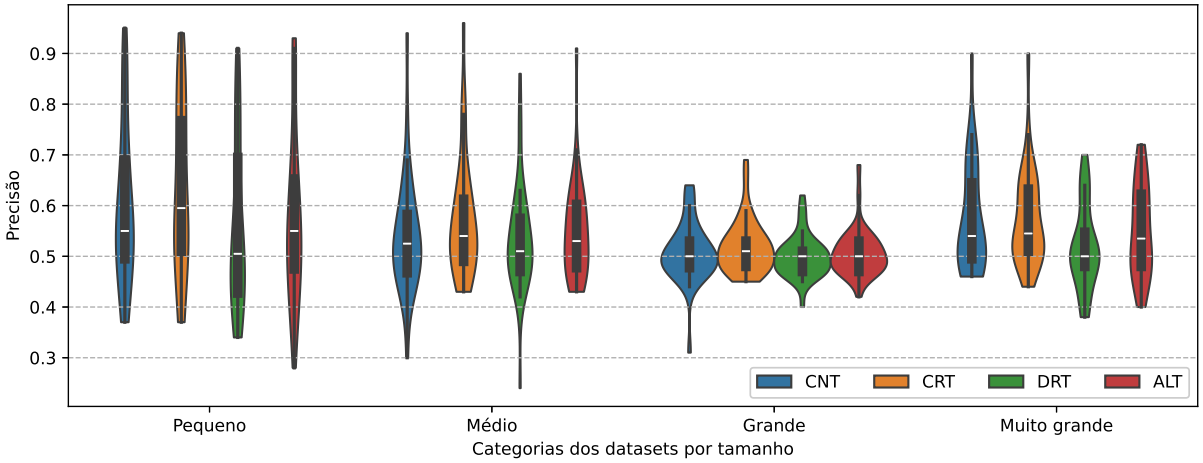
Fonte: Elaborada pelo autor.

Figura 44 – Sensibilidade do modelo de acordo com o tamanho das bases de dados.



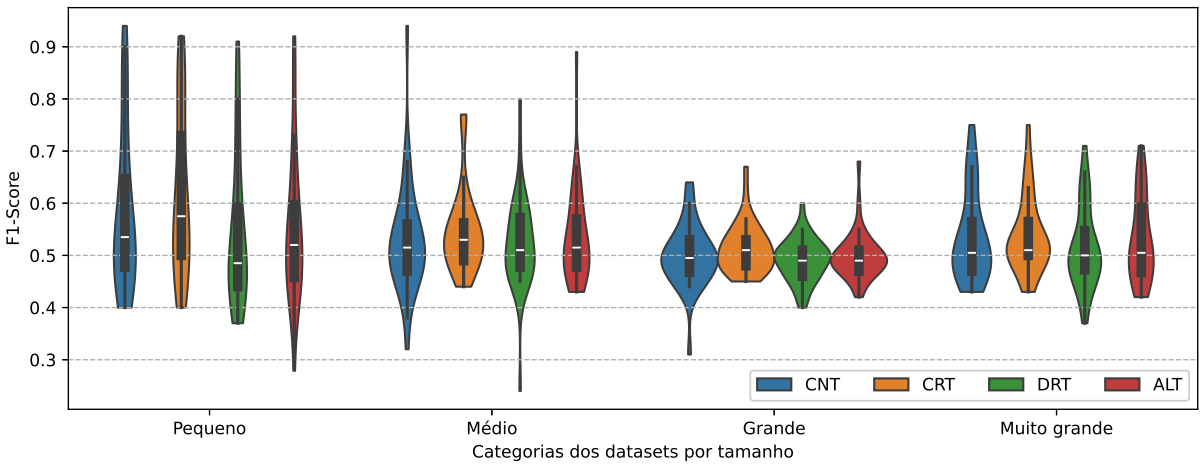
Fonte: Elaborada pelo autor.

Figura 45 – Precisão do modelo de acordo com o tamanho das bases de dados.



Fonte: Elaborada pelo autor.

Figura 46 – F1-score do modelo de acordo com o tamanho das bases de dados.



Fonte: Elaborada pelo autor.

Em síntese, observa-se que o desempenho global do modelo melhora com o aumento do tamanho do conjunto de dados, ao mesmo tempo em que revela diferenças significativas entre as categorias analisadas, reforçando a importância tanto da escala dos dados quanto da natureza do conjunto utilizado na avaliação do modelo.

5 Conclusão

Este trabalho propôs e desenvolveu um algoritmo construtivo para redes neurais totalmente conectadas do tipo *stacked autoencoder*, baseado em uma estratégia de crescimento progressivo da arquitetura durante o treinamento, sem a necessidade de definir previamente o tamanho final da rede. Diferentemente de algumas abordagens tradicionais, que partem de arquiteturas fixas e frequentemente superdimensionadas, o método adota uma lógica incremental, adicionando novas camadas apenas quando há evidência de ganho no processo de aprendizado. Essa estratégia permitiu controlar o aumento da complexidade do modelo e analisar, de forma sistemática, o impacto das modificações estruturais no desempenho.

A análise dos diferentes tipos de inserção de novas camadas (constante, crescente, decrescente e alaeatória) demonstrou que o padrão de crescimento exerce influência direta tanto no desempenho quanto na estabilidade do aprendizado. A comparação entre o ajuste dos pesos das conexões das camadas de saída de cada ramo e a existência de um ramo extra evidenciou que maior liberdade de ramificação não implica, necessariamente, melhor desempenho. Os resultados indicam que abordagens com ajuste dos pesos entre as saídas de cada ramo apresentam maior consistência nas métricas classificatórias, enquanto que a inclusão de ramo extra mostrou maior variabilidade, sugerindo que estratégias de crescimento mais controladas são mais adequadas em cenários práticos. Assim, observou-se que as abordagens que incorporam o crescimento de neurônios na etapa de inserção, o uso de ramos extras e o ajuste dos pesos de conexão entre as saídas de cada ramo ampliam de forma consistente a capacidade de exploração do espaço de soluções. Nesse contexto, os resultados evidenciam de maneira clara a superioridade do método de inserção crescente, que apresentou o melhor desempenho global ao longo dos experimentos. Em seguida, o método de inserção constante também demonstrou resultados satisfatórios, ainda que inferiores, confirmando sua eficácia como estratégia intermediária. Esses achados reforçam a importância de mecanismos de expansão progressiva na construção da rede, contribuindo diretamente para ganhos de desempenho e robustez do modelo.

Nos experimentos com dados de áudio, o método foi avaliado na classificação de sílabas fonéticas da língua portuguesa, permitindo verificar seu comportamento em um domínio distinto das bases tabulares. Mesmo diante das particularidades dessa tarefa, o algoritmo manteve desempenho consistente, sem ocorrência de problemas relacionados à qualidade do sinal, demonstrando sua robustez e adaptabilidade.

A validação em múltiplas bases do OpenML-CC18 confirmou a capacidade de generalização do método frente a conjuntos de dados heterogêneos e diferentes distribuições de classes. A análise das métricas de acurácia, precisão, sensibilidade e especificidade mostrou vantagens do modelo proposto em relação às abordagens tradicionais, por usar menos camadas e sofrer

crescimento apenas quando há benefícios comprovados. Esse comportamento reduz o custo computacional e o esforço de projeto associados a arquiteturas profundas definidas *a priori*.

Outro aspecto relevante é a preservação do conhecimento previamente adquirido. O método mostrou ser capaz de estender a arquitetura sem comprometer o aprendizado já realizado, evitando reinicializações completas e minimizando perdas de desempenho, o que reduz desperdício computacional e dependência de múltiplas tentativas de configuração arquitetural, culminando na redução do tempo de projeto e na redução do consumo de energia.

De forma geral, os resultados indicam que a abordagem atinge seu objetivo principal: construir redes neurais de forma adaptativa, com menor intervenção manual e maior eficiência no uso da capacidade do modelo. A pesquisa evidencia que métodos construtivos podem ser mais competitivos que arquiteturas fixas e, em muitos cenários, superá-las justamente por explorarem de forma mais eficiente o crescimento incremental.

5.1 Limitações do trabalho

Apesar dos resultados promissores, o trabalho apresenta limitações que precisam ser consideradas. O critério de inserção de novos ramos ainda depende de limiares definidos manualmente, como o número máximo de épocas e o erro mínimo aceitável. Isso exige conhecimento prévio do problema e pode comprometer a generalidade do método, principalmente em cenários menos explorados ou com pouca informação disponível.

Além disso, a avaliação experimental concentrou-se em tarefas de classificação com dados tabulares e de áudio. Outros domínios importantes, como séries temporais, imagens e dados textuais, não foram contemplados, o que indica que a eficácia da abordagem construtiva proposta ainda precisa ser investigada nesses contextos.

Por fim, o método foi desenvolvido e validado exclusivamente para redes totalmente conectadas do tipo stacked autoencoder. Dessa forma, sua aplicação direta a arquiteturas convolucionais ou recorrentes não é imediata, sendo necessárias adaptações estruturais adicionais para viabilizar sua extensão a esses modelos.

5.2 Trabalhos futuros

Como trabalhos futuros, destacam-se:

- Incorporação de critérios automáticos de parada do crescimento: definição de métricas objetivas (como ganho marginal em validação, redução do erro ou estabilidade do gradiente) para interromper a inserção de novos ramos quando o acréscimo estrutural da rede deixar de produzir melhorias significativas;

- Pesos nulos anteriores à nova camada escondida: inicializar os pesos anteriores à camada escondida do novo ramo com valores nulos e os pesos posteriores com valores aleatórios, além de usar a função tanh como função de ativação dos neurônios ocultos dessa camada, de modo a acelerar o aprendizado e ainda garantir a colaboração do novo ramo;
- Poda de neurônios ao longo das etapas construtivas: aplicação de estratégias de remoção de neurônios com baixa contribuição para a saída final, promovendo compactação do modelo e maior eficiência computacional durante e após o crescimento incremental;
- Avaliação do método em aplicações que demandam adaptação contínua: análise do comportamento da arquitetura em cenários dinâmicos, como fluxos de dados não estacionários ou ambientes de aprendizado contínuo, verificando sua capacidade de expansão controlada, estabilidade e retenção de conhecimento previamente adquirido.
- Perspectiva de extensão do algoritmo para outras arquiteturas de redes neurais, mais especificamente em redes convolucionais.

Referências

ACADEMY, Data Science. **Deep Learning Book**. 2021. URL: <https://www.deeplearningbook.com.br>. Último acesso em 20/11/2021. Nenhuma citação no texto.

AGARWAL, Akhil; ATHALYE, Anish. Stacking layers for accelerated gradient descent. **arXiv preprint arXiv:2408.08011**, 2024. Disponível em: <https://arxiv.org/abs/2408.08011>. Nenhuma citação no texto.

AGHILI, Joubine; MULA, Olga. Depth-adaptive neural networks from the optimal control viewpoint. **arXiv preprint arXiv:2007.02428**, 2020. Preprint. Disponível em: <https://arxiv.org/abs/2007.02428>. Nenhuma citação no texto.

ALBERTI, Michele *et al.* A pitfall of unsupervised pre-training. **arXiv preprint arXiv:1703.04332**, 2017. Nenhuma citação no texto.

ALFAYEZ, Sarah; BCHIR, Ouiem; ISMAIL, Mohamed Maher Ben. Dynamic depth learning in stacked autoencoders. **Applied Sciences**, v. 13, n. 19, p. 10994, 2023. Nenhuma citação no texto.

ASHFAHANI, Andri *et al.* Devdan: Deep evolving denoising autoencoder. **Neurocomputing**, Elsevier, v. 390, p. 297–314, 2020. Nenhuma citação no texto.

AURÉLIEN, Géron. Hands-on machine learning with scikit-learn & tensorflow. **Geron Aurelien**, 2017. Nenhuma citação no texto.

BADRINARAYANAN, Vijay; KENDALL, Alex; CIPOLLA, Roberto. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 39, n. 12, p. 2481–2495, 2017. Nenhuma citação no texto.

BANK, Dor; KOENIGSTEIN, Noam; GIRYES, Raja. Autoencoders. **arXiv preprint arXiv:2003.05991**, 2020. Nenhuma citação no texto.

BATISTA, Cassio; DIAS, Ana Larissa; NETO, Nelson. Free resources for forced phonetic alignment in brazilian portuguese based on kaldi toolkit. **EURASIP Journal on Advances in Signal Processing**, Springer, v. 2022, n. 1, p. 11, 2022. Disponível em: <https://doi.org/10.1186/s13634-022-00844-9>. Nenhuma citação no texto.

BATISTA, Cassio; NETO, Nelson. Forced phonetic alignment in brazilian portuguese using time-delay neural networks. In: SPRINGER. **International Conference on Computational Processing of the Portuguese Language**. 2022. p. 323–332. Disponível em: <https://doi.org/10.1007/978-3-030-98305-5_30>. Nenhuma citação no texto.

BENGIO, Yoshua. **Learning Deep Architectures for AI**. [S.l.]: Foundations and Trends in Machine Learning, 2009. Nenhuma citação no texto.

BENGIO, Yoshua; COURVILLE, Aaron. Representation learning revisited. **Foundations and Trends in Machine Learning**, v. 14, n. 4, p. 249–392, 2021. Nenhuma citação no texto.

BENGIO, Yoshua; COURVILLE, Aaron; VINCENT, Pascal. Representation learning: A review and new perspectives. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 35, n. 8, p. 1798–1828, 2013. Nenhuma citação no texto.

BERAHMAND, Kamal *et al.* Autoencoders and their applications in machine learning: a survey. **Artificial Intelligence Review**, v. 57, n. 2, p. 28, 2024. Nenhuma citação no texto.

BISCHL, Bernd *et al.* Openml benchmarking suites. **arXiv:1708.03731v2 [stat.ML]**, 2019. Nenhuma citação no texto.

BOUTARFASS, Sanae; BESSERER, Bernard. Convolutional autoencoder for discriminating handwriting styles. In: IEEE. **2019 8th European Workshop on Visual Information Processing (EUVIP)**. [S.l.], 2019. p. 199–204. Nenhuma citação no texto.

BRAGA, Antônio de Pádua; CARVALHO, André Ponce de Leon F. de; LUDERMIR, Teresa Bernarda. **Redes Neurais Artificiais: Teoria e Aplicações**. 1. ed. Rio de Janeiro: LTC, 2000. Nenhuma citação no texto.

BUDUMA, N.; LACASCIO, N. **Fundamentals of Deep Learning**. 1. ed. [S.l.]: O'REILLY, 2017. Nenhuma citação no texto.

CAILLON, Pierre; CERISARA, Frédéric. Growing neural networks: A theoretical perspective. **Neural Networks**, v. 141, p. 1–15, 2021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893608021000844>. Nenhuma citação no texto.

CAO, Mingzi; WANG, Xi; ALETRAS, Nikolaos. Progressive depth up-scaling via optimal transport. **arXiv preprint arXiv:2508.08011**, 2025. Disponível em: <https://arxiv.org/abs/2508.08011>. Nenhuma citação no texto.

CHEN, Jinghui *et al.* Outlier detection with autoencoder ensembles. In: SIAM. **Proceedings of the 2017 SIAM international conference on data mining**. [S.l.], 2017. p. 90–98. Nenhuma citação no texto.

CHEN, X.; AL. et. Efficient autoencoder design through adaptive layer expansion. **IEEE Transactions on Neural Networks and Learning Systems**, 2024. Nenhuma citação no texto.

DANTAS, Leonardo Camilo. Otimização de redes neurais artificiais de múltiplas camadas utilizando algoritmos genéticos e enxame de partículas. 2017. Nenhuma citação no texto.

DENG, Li; YU, Dong. Deep learning: methods and applications. **Foundations and trends in signal processing**, Now Publishers Inc. Hanover, MA, USA, v. 7, n. 3–4, p. 197–387, 2014. Nenhuma citação no texto.

DIAS, Ana Larissa *et al.* Towards a free, forced phonetic aligner for brazilian portuguese using kaldi tools. In: SPRINGER. **Brazilian Conference on Intelligent Systems**. 2020. p. 621–635. Disponível em: <https://doi.org/10.1007/978-3-030-61377-8_44>. Nenhuma citação no texto.

DOERSCH, Carl. Tutorial on variational autoencoders. **arXiv preprint arXiv:1606.05908**, 2016. Nenhuma citação no texto.

DONG, Bo; AL. et. Deep autoencoder architectures for representation learning. **Information Sciences**, v. 624, p. 1–23, 2023. Nenhuma citação no texto.

DONG, Ganggang *et al.* A review of the autoencoder and its variants: A comparative perspective from target recognition in synthetic-aperture radar images. **IEEE Geoscience and Remote Sensing Magazine**, v. 6, n. 3, p. 44–68, 2018. Nenhuma citação no texto.

ELSKEN, Thomas; METZEN, Jan Hendrik; HUTTER, Frank. Neural architecture search: A survey. **Journal of Machine Learning Research**, v. 23, p. 1–72, 2022. Nenhuma citação no texto.

EVCI, Utku *et al.* Gradmax: Growing neural networks using gradient information. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2022. Nenhuma citação no texto.

FAHLMAN, Scott E.; LEBIERE, Christian. The cascade-correlation learning architecture. In: **Advances in Neural Information Processing Systems (NeurIPS)**. [S.l.: s.n.], 1990. p. 524–532. Trabalho fundador das redes neurais construtivas, introduzindo o conceito de crescimento incremental de neurônios. Nenhuma citação no texto.

FALABRASIL, Grupo. **Speech Datasets – Male/Female (MF) for Forced Phonetic Alignment**. 2023. <https://github.com/falabrasil/speech-datasets>. Universidade Federal do Pará. Nenhuma citação no texto.

FEHRING, Johannes; SCHMIDHUBER, Jürgen. Growing deep reinforcement learning agents. **arXiv preprint arXiv:2508.08011**, 2025. Disponível em: <https://arxiv.org/abs/2508.08011>. Nenhuma citação no texto.

FIESLER, Emile. Comparative bibliography of ontogenic neural networks. In: CITESEER. **Proceedings of the International Conference on Artificial Neural Networks**. [S.l.], 1994. v. 1, p. 793–796. Nenhuma citação no texto.

FREAN, Marcus. The upstart algorithm: A method for constructing and training feedforward neural networks. **Neural computation**, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 2, n. 2, p. 198–209, 1990. Nenhuma citação no texto.

FRITZKE, Bernd. A growing neural gas network learns topologies. **Advances in Neural Information Processing Systems**, v. 7, p. 625–632, 1995. Nenhuma citação no texto.

GALLANT, Stephen I; GALLANT, Stephen I. **Neural network learning and expert systems**. [S.l.]: MIT press, 1993. Nenhuma citação no texto.

GALLANT, Stephen I *et al.* Perceptron-based learning algorithms. **IEEE Transactions on neural networks**, v. 1, n. 2, p. 179–191, 1990. Nenhuma citação no texto.

GAMBELLA, C.; AL. et. Seal: Scalable evolutionary architecture learner. **Early Access**, 2025. Nenhuma citação no texto.

GAO, Qiang; LUO, Zhipeng; KLABJAN, Diego. Efficient architecture search for continual learning. **arXiv preprint arXiv:2006.04027**, 2020. Nenhuma citação no texto.

GERON, Aurelion. **Hands-On Machine Learning with Scikit-Learn and TensorFlow**. 1. ed. [S.l.]: O'Reilly, 2017. Nenhuma citação no texto.

GHOSH, Joydeep; TUMER, Kagan. Structural adaptation and generalization in supervised feedforward networks. **Journal of Artificial Neural Networks**, Citeseer, v. 1, n. 4, p. 431–458, 1994. Nenhuma citação no texto.

GONDARA, Lovedeep. Medical image denoising using convolutional denoising autoencoders. In: IEEE. **2016 IEEE 16th international conference on data mining workshops (ICDMW)**. [S.l.], 2016. p. 241–246. Nenhuma citação no texto.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. [S.l.]: MIT Press, 2016. Nenhuma citação no texto.

HAN, J.; AL. et. Deep self-developing neural networks. **Pattern Recognition**, v. 144, p. 108719, 2023. Nenhuma citação no texto.

HASSIBI, Babak; STORK, David G. Second order derivatives for network pruning: optimal brain surgeon. **Advances in Neural Information Processing Systems**, v. 5, p. 164–171, 1993. Nenhuma citação no texto.

HAY, Jonathan; RUDER, Sebastian. Dynamic layer tying for parameter-efficient transformers. **arXiv preprint arXiv:2401.12819**, 2024. Disponível em: <https://arxiv.org/abs/2401.12819>. Nenhuma citação no texto.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. 2. ed. [S.l.]: Bookman, 2001. Nenhuma citação no texto.

HAYKIN, Simon. **Neural networks and learning machines, 3/E**. [S.l.]: Pearson Education India, 2009. Nenhuma citação no texto.

HEUILLET, A.; BELAID, A. Dynamic architecture search: A survey. **Engineering Applications of AI**, v. 118, p. 104126, 2023. Nenhuma citação no texto.

HINTON, Geoffrey E; SALAKHUTDINOV, Ruslan R. Reducing the dimensionality of data with neural networks. **Science**, v. 313, n. 5786, p. 504–507, 2006. Nenhuma citação no texto.

HONORATO, Eduardo; MUNIZ, Victor da Silva; FILHO, Joao Souza. Detecção hierárquica de classes desconhecidas em sonar por "autoencoders"convolucionais. In: . [S.l.: s.n.], 2020. Nenhuma citação no texto.

HORNIK, Kurt; STINCHCOMBE, Maxwell; WHITE, Halbert. Multilayer feedforward networks are universal approximators. **Neural networks**, Elsevier, v. 2, n. 5, p. 359–366, 1989. Nenhuma citação no texto.

HUANG, Shenyang; FRANçOIS-LAVET, Vincent; RABUSSEAU, Guillaume. Neural architecture search for class-incremental learning. **arXiv preprint arXiv:1909.06686**, 2019. Nenhuma citação no texto.

HUBENS, Nathan. **Deep inside: Autoencoders**. 2018. URL: <https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f>. Último acesso em 20/11/2021. Nenhuma citação no texto.

HWANG, Jeng-Neng *et al.* Regression modeling in back-propagation and projection pursuit learning. **IEEE Transactions on neural networks**, IEEE, v. 5, n. 3, p. 342–353, 1994. Nenhuma citação no texto.

JIANG, Yiding *et al.* Fantastic generalization measures and where to find them. **ICLR**, 2020. Nenhuma citação no texto.

KALMANOVICH, Alexander; CHECHIK, Gal. Gradual training of deep denoising autoencoders. In: **Proceedings of the 31st International Conference on Machine Learning (ICML)**. [s.n.], 2014. p. 1795–1803. Disponível em: <https://arxiv.org/abs/1412.6257>. Nenhuma citação no texto.

KAPLUN, Gal; SOUDRY, Daniel; MEIR, Ron. Predicting generalization in deep learning via deep kernels. **ICLR**, 2022. Nenhuma citação no texto.

KINGMA, Diederik P; WELLING, Max. An introduction to variational autoencoders. **arXiv preprint arXiv:1906.02691**, 2019. Nenhuma citação no texto.

KOZAL, A.; SENER, O.; KOLTUN, V. Increasing neural network depth in continual learning. **Neurocomputing**, v. 491, p. 245–258, 2022. Nenhuma citação no texto.

LAHNAJÄRVI, Jani JT; LEHTOKANGAS, Mikko I; SAARINEN, Jukka PP. Evaluation of constructive neural networks with cascaded architectures. **Neurocomputing**, Elsevier, v. 48, n. 1-4, p. 573–607, 2002. Nenhuma citação no texto.

LECUN, Yann; DENKER, John S.; SOLLA, Sara A. Optimal brain damage. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 1990. v. 2, p. 598–605. Nenhuma citação no texto.

LIN, Shaobo; ZENG, Jinshan; ZHANG, Xiaoqin. Constructive neural network learning. **arXiv preprint arXiv:1601.00144**, 2016. Propõe formalização teórica moderna para aprendizado construtivo em redes neurais. Nenhuma citação no texto.

MAJUMDAR, Angshul; TRIPATHI, Aditay. Asymmetric stacked autoencoder. In: IEEE. **2017 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2017. p. 911–918. Nenhuma citação no texto.

MAKHZANI, Alireza; FREY, Brendan. K-sparse autoencoders. **arXiv preprint arXiv:1312.5663**, 2013. Nenhuma citação no texto.

MICONI, Thomas. Neural networks with differentiable structure. **arXiv preprint arXiv:1606.06216**, 2016. Nenhuma citação no texto.

NI, Z.; AL. et. Adaptive layer-adding deep autoencoders. **Neurocomputing**, 2024. Nenhuma citação no texto.

PAN, Xinglin. Growing neural networks using orthogonal initialization. In: **Third International Seminar on Artificial Intelligence, Networking, and Information Technology**. [S.l.: s.n.], 2023. Nenhuma citação no texto.

PEDRAZA, Anibal *et al.* Leveraging autoencoders and chaos theory to improve adversarial example detection. **Neural Computing and Applications**, v. 36, n. 10, p. 18265–18275, 2024. Nenhuma citação no texto.

PEDRAZA, C.; AL. et. Self-growing neural systems for high-dimensional learning. **Expert Systems with Applications**, v. 220, p. 119619, 2024. Nenhuma citação no texto.

PEREIRA, Bianca Valéria Lopes. **Reconhecimento de fonemas com compactação das frequências via centroide e redes stacked autoencoders.** 2014. Dissertação (Mestrado) — Universidade Federal do Maranhão, São Luis - MA. Disponível em: <https://tedebc.ufma.br/jspui/handle/tede/5486>. Nenhuma citação no texto.

PIMENTEL, Marco AF *et al.* A review of novelty detection. **Signal Processing**, Elsevier, v. 99, p. 215–249, 2014. Nenhuma citação no texto.

RADHAKRISHNAN, Anil *et al.* Growing neural networks: dynamic evolution through gradient descent. **Proceedings of the Royal Society A**, 2025. Nenhuma citação no texto.

REED, Russell. Pruning algorithms-a survey. **IEEE transactions on Neural Networks**, IEEE, v. 4, n. 5, p. 740–747, 1993. Nenhuma citação no texto.

RUSSO, Stefania *et al.* Anomaly detection using deep autoencoders for in-situ wastewater systems monitoring data. **arXiv preprint arXiv:2002.03843**, 2020. Nenhuma citação no texto.

RUSU, Andrei A. *et al.* Progressive neural networks. **arXiv preprint arXiv:1606.04671**, 2016. Nenhuma citação no texto.

SCACCIA, Kevin. **Redes Neurais Artificiais – Introdução à Autoencoders**. 2020. URL: <https://dataml.com.br/introducao-a-autoencoders/>. Último acesso em 20/11/2021. Nenhuma citação no texto.

SHARIR, Or; AL. et. Incremental neural networks. **ICML**, 2023. Nenhuma citação no texto.

SILVA, Ivan Nunes da; SPATTI, Danilo Hernane; FLAUZINO, Rogério Andrade. **Redes Neurais Artificiais para Engenharia e Ciências Aplicadas: Fundamentos Teóricos e Aspectos Práticos**. 2. ed. São Paulo: Artliber, 2016. Nenhuma citação no texto.

SMITH, Leslie N.; HAND, Emily M.; DOSTER, Timothy. Gradual dropin of layers to train very deep neural networks. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [s.n.], 2015. p. 5978–5986. Disponível em: <https://ieeexplore.ieee.org/document/7299076>. Nenhuma citação no texto.

SØNDERBY, Casper Kaae *et al.* How to train deep variational autoencoders and probabilistic ladder networks. **arXiv preprint arXiv:1602.02282**, v. 3, 2016. Nenhuma citação no texto.

SOUZA, Gleidson; NETO, Nelson. An automatic phonetic aligner for brazilian portuguese with a praat interface. In: SPRINGER. **Computational Processing of the Portuguese Language: 12th International Conference, PROPOR 2016, Tomar, Portugal, July 13-15, 2016, Proceedings 12**. 2016. p. 374–384. Disponível em: <https://doi.org/10.1007/978-3-319-41552-9_38>. Nenhuma citação no texto.

SRIVASTAVA, Nitish *et al.* Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, 2014. Nenhuma citação no texto.

SUN, Kai *et al.* Generalized extreme learning machine autoencoder and a new deep neural network. **Neurocomputing**, Elsevier, v. 230, p. 374–381, 2017. Nenhuma citação no texto.

TURNER, Daniel; CARDOSO, Pedro J. S.; RODRIGUES, J. M. F. Modular dynamic neural network: A continual learning architecture. **Applied Sciences**, v. 11, n. 24, p. 12078, 2021. Nenhuma citação no texto.

ULLAH, Amin *et al.* Action recognition using optimized deep autoencoder and cnn for surveillance data streams of non-stationary environments. **Future Generation Computer Systems**, Elsevier, v. 96, p. 386–397, 2019. Nenhuma citação no texto.

VIANA, Francisco dos Santos. **Redes Neurais Aplicadas na Estratégia de Ponderação de Partículas em SLAM**. 2019. Mestrado Profissional em Engenharia da Computação — Universidade Estadual do Maranhão, Maranhão. Nenhuma citação no texto.

VILLANUEVA, Wilfredo Jaime Puma. **Síntese automática de redes neurais artificiais com conexões à frente arbitrárias**. 2011. Tese (Doutorado) — Tese de Doutorado, Faculdade de Engenharia Elétrica e Computação, Campinas .... Nenhuma citação no texto.

VINCENT, Pascal *et al.* Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. **Journal of Machine Learning Research**, v. 11, p. 3371–3408, 2010. Nenhuma citação no texto.

WANG, R.; AL. et. Adaptive deep networks: Progress and challenges. **Pattern Recognition**, 2024. Nenhuma citação no texto.

WANG, Xiaoyu; ZHANG, Yifan; LIU, Wei. Dynamic layer attention for deep neural networks. **arXiv preprint arXiv:2406.13392**, 2024. Disponível em: <https://arxiv.org/abs/2406.13392>. Nenhuma citação no texto.

WHITE, Colin; ZELA, Andreas; AL. et. Nas-bench-301 and the future of nas benchmarks. **NeurIPS**, 2021. Nenhuma citação no texto.

WISTUBA, M. A survey of hyperparameter optimization and neural architecture search in deep learning. **ACM Computing Surveys**, v. 56, n. 2, p. 1–35, 2023. Nenhuma citação no texto.

WOLBERG, WIlliam. **Breast Cancer Wisconsin (Original)**. 1992. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5HP4Z. Nenhuma citação no texto.

XU, Zhiqiang *et al.* Progressive learning: A deep learning framework for continual learning. **Neural Networks**, v. 128, p. 345–357, 2020. Nenhuma citação no texto.

YAN, J.; AL. et. Deep evolving representation networks. **Neurocomputing**, v. 437, p. 1–16, 2021. Nenhuma citação no texto.

YANG, Jian; XIONG, Zhiwei; ZHU, Jun. Lesa: Learnable llm layer scaling-up. **arXiv preprint arXiv:2502.13794**, 2025. Disponível em: <https://arxiv.org/abs/2502.13794>. Nenhuma citação no texto.

YE, X.; AL. et. Dynamic expansion mechanisms for continual learning. **IEEE Transactions on Neural Networks and Learning Systems**, 2023. Nenhuma citação no texto.

YUAN, X.; AL. et. Firefly-inspired constructive neural networks. **Neurocomputing**, v. 543, p. 1–15, 2023. Nenhuma citação no texto.

ZHANG, Chiyuan *et al.* Understanding deep learning requires rethinking generalization (revisited). **Journal of Machine Learning Research**, v. 22, p. 1–79, 2021. Nenhuma citação no texto.

ZHANG, Xinlin *et al.* Review of deep neural network based on auto-encoder. **DEStech Transactions on Computer Science and Engineering**, n. iciti, 2018. Nenhuma citação no texto.

ZHU, Xiaoyang; XIE, Lingxi; TIAN, Qi. Adaptive growth: Real-time cnn layer expansion. **arXiv preprint arXiv:2309.03049**, 2023. Disponível em: <https://arxiv.org/abs/2309.03049>. Nenhuma citação no texto.

ZHUANG, Fuzhen *et al.* Supervised representation learning: Transfer learning with deep autoencoders. In: **Twenty-Fourth International Joint Conference on Artificial Intelligence**. [S.l.: s.n.], 2015. Nenhuma citação no texto.

ZUBEN, Fernando José Von. **Modelos parametricos e não-parametricos de redes neurais artificiais e aplicações.** 1996. Tese (Doutorado) — University of Campinas, Brazil. Nenhuma citação no texto.

A Interface Gráfica do *Front-end* RNA-Colaborativa

Este apêndice descreve a interface gráfica do *front-end* da ferramenta **Rede Neural Colaborativa – CollabNet**, desenvolvida para configurar, treinar e avaliar redes neurais incrementais baseadas na arquitetura colaborativa proposta. A interface é organizada em dois painéis principais: o painel de controle (esquerda/centro) e o painel de gráficos (direita), cada um com abas específicas detalhadas nas seções a seguir.

A.1 Visão Geral da Interface

A janela principal é composta por três regiões funcionais:

- **Painel de Controle** (coluna esquerda): contém as abas *Configurações* e *Métricas de Desempenho*, responsáveis pela parametrização do treinamento e pela definição das métricas de avaliação;
- **Painel de Resumo** (coluna central): exibe o contador de camadas adicionadas e o histórico de execução durante o processo de treinamento;
- **Painel de Gráficos** (coluna direita): apresenta visualizações em tempo real do treinamento e avaliação, organizado nas abas *Gráficos do Treinamento*, *Gráfico da Saída da RNA* e *Gráficos de Avaliação*.

A.2 Aba *Configurações*

A aba *Configurações*, ilustrada na Figura 47, reúne todos os controles necessários para parametrizar e executar o treinamento da rede neural. Ela é subdividida nos seguintes grupos:

A.2.1 Parâmetros de Treinamento

Este grupo define os hiperparâmetros globais da rede, conforme descrito na Tabela 14.

A.2.2 Funções de Ativação

Três grupos de funções de ativação podem ser configurados independentemente:

- **Função da Camada Oculta**: define a função de ativação dos neurônios da camada oculta de cada ramo. O campo numérico adjacente permite ajustar o parâmetro de

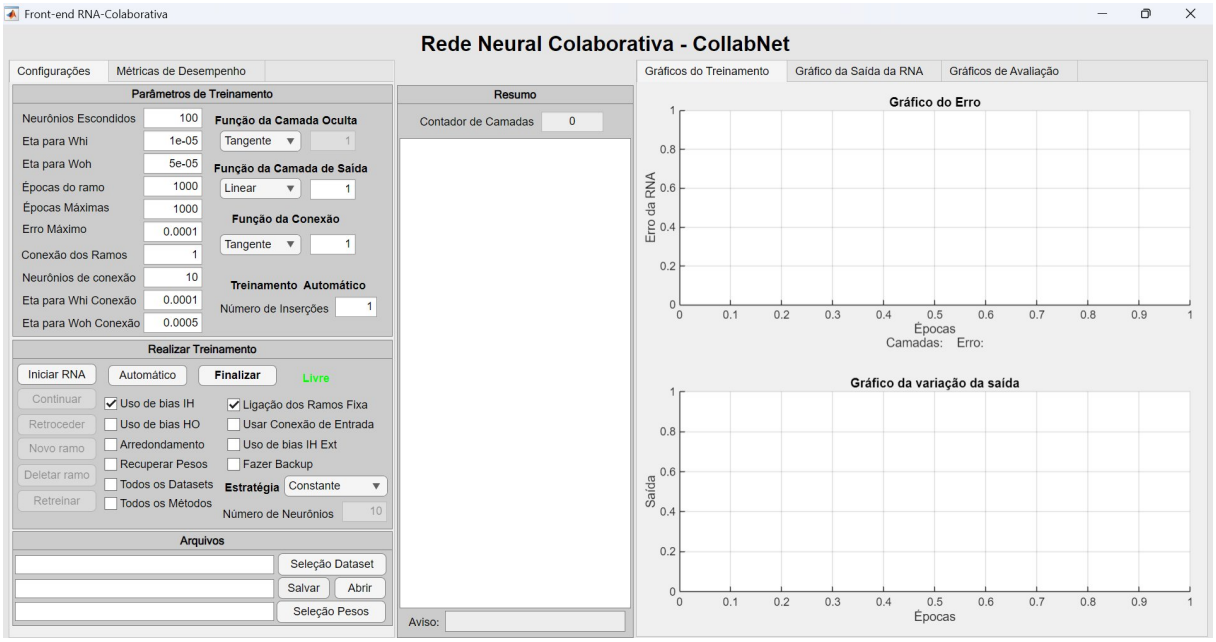


Figura 47 – Aba *Configurações* da interface CollabNet.

Tabela 14 – Parâmetros de treinamento e seus significados.

Parâmetro	Descrição
Neurônios Escondidos	Quantidade de neurônios na camada oculta de cada ramo adicionado à rede.
Eta para Whi	Taxa de aprendizagem para os pesos da camada de entrada para a camada oculta (η_{Whi}).
Eta para Woh	Taxa de aprendizagem para os pesos da camada oculta para a camada de saída (η_{Woh}).
Épocas do Ramo	Número máximo de épocas de treinamento por ramo inserido.
Épocas Máximas	Limite global de épocas para todo o processo de treinamento.
Erro Máximo	Critério de parada por erro: o treinamento é interrompido quando o erro atinge esse limiar.
Conexão dos Ramos	Número de ramos anteriores cujas saídas são utilizadas como entradas adicionais ao ramo atual (profundidade da conexão colaborativa).
Neurônios de Conexão	Quantidade de neurônios utilizados na camada de conexão entre ramos.
Eta para Whi Conexão	Taxa de aprendizagem para os pesos da camada de conexão ($\eta_{Whi,\text{con}}$).
Eta para Woh Conexão	Taxa de aprendizagem para os pesos de saída da camada de conexão ($\eta_{Woh,\text{con}}$).

inclinação da função selecionada. As opções disponíveis incluem *Tangente* (tangente hiperbólica), *Sigmoide* e *Linear*.

- **Função da Camada de Saída**: define a função de ativação da camada de saída da rede. O padrão é *Linear*, adequado para tarefas de regressão.
- **Função da Conexão**: define a função de ativação utilizada nos neurônios da camada de conexão entre ramos colaborativos.

A.2.3 Treinamento Automático

O campo **Número de Inserções** define quantos ramos serão adicionados sequencialmente de forma automática ao acionar o botão *Automático*. Isso permite executar múltiplos ciclos de inserção e treinamento sem intervenção manual a cada etapa.

A.2.4 Realizar Treinamento

Este grupo concentra os controles de execução do processo de treinamento:

- **Iniciar RNA**: inicializa a rede neural com os parâmetros configurados e treina o primeiro ramo;
- **Automático**: executa o número de inserções definido automaticamente, adicionando e treinando ramos em sequência;
- **Finalizar**: encerra o processo de treinamento;
- **Continuar**: retoma um treinamento pausado;
- **Retroceder**: remove o último ramo adicionado;
- **Novo Ramo**: adiciona manualmente um novo ramo e o treina;
- **Deletar Ramo**: exclui o ramo selecionado;
- **Retreinar**: re-executa o treinamento do ramo atual.

As caixas de seleção permitem habilitar ou desabilitar opções como uso de *bias* nas camadas, arredondamento de pesos, recuperação de pesos salvos, *backup* automático, uso de conexão de entrada e uso de *bias* externo. O campo **Estratégia** define a política de alocação de neurônios ao longo do treinamento (*Constante*, *Crescente*, etc.), e o campo **Número de Neurônios** indica a quantidade utilizada na estratégia selecionada.

A.2.5 Arquivos

A seção de arquivos oferece três operações principais:

- **Seleção Dataset**: carrega o arquivo de dados de treinamento;
- **Salvar / Abrir**: salva ou carrega uma configuração completa da rede (pesos e parâmetros);
- **Seleção Pesos**: carrega um arquivo de pesos previamente exportado.

A.3 Aba *Métricas de Desempenho*

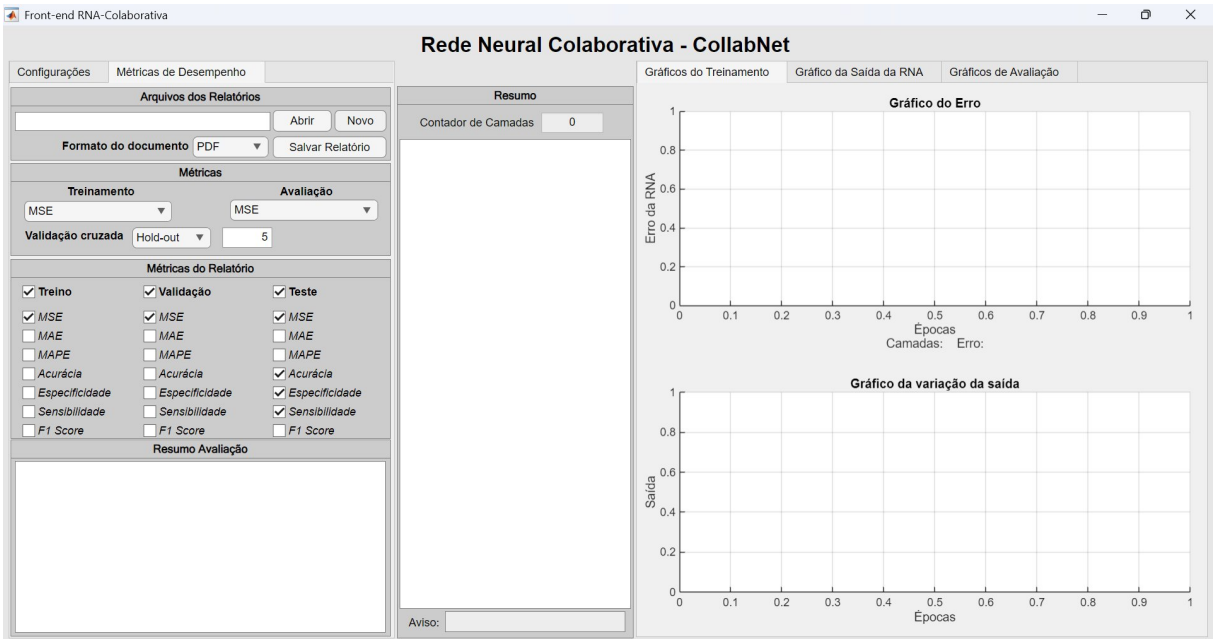


Figura 48 – Aba *Métricas de Desempenho* da interface CollabNet.

A aba *Métricas de Desempenho*, apresentada na Figura 48, permite configurar como os resultados de treinamento e avaliação são calculados e reportados.

A.3.1 Arquivos dos Relatórios

Os controles desta seção permitem abrir um relatório existente (*Abrir*), criar um novo (*Novo*), definir o formato do documento exportado (PDF, HTML, entre outros) e salvar o relatório gerado (*Salvar Relatório*).

A.3.2 Métricas de Treinamento e Avaliação

Dois menus suspensos independentes permitem selecionar a métrica de erro utilizada durante o treinamento e a métrica empregada na avaliação final. As opções incluem **MSE** (*Mean Squared Error*), **MAE** (*Mean Absolute Error*) e **MAPE** (*Mean Absolute Percentage Error*).

A.3.3 Validação Cruzada

Define o método de particionamento dos dados para avaliação do modelo. O menu **Validação Cruzada** oferece opções como *Hold-out* e *K-Fold*, e o campo numérico adjacente configura o parâmetro correspondente (proporção de teste no *hold-out* ou valor de k no *K-Fold*).

A.3.4 Métricas do Relatório

Esta seção permite escolher quais métricas serão incluídas no relatório gerado, organizadas em três colunas correspondentes às partições dos dados: **Treino**, **Validação** e **Teste**. As métricas disponíveis em cada partição são:

- **MSE**, **MAE**, **MAPE**: métricas de erro para tarefas de regressão;
- **Acurácia**, **Especificidade**, **Sensibilidade**, **F1 Score**: métricas para tarefas de classificação.

A.3.5 Resumo de Avaliação

Área de texto que exibe, ao final do treinamento, um resumo consolidado dos valores calculados para as métricas selecionadas nas três partições.

A.4 Aba *Gráficos do Treinamento*

A aba *Gráficos do Treinamento* (visível nas Figuras 47 e 48) exibe dois gráficos atualizados em tempo real durante o processo de treinamento:

- **Gráfico do Erro**: apresenta a evolução do erro da RNA ao longo das épocas para cada ramo treinado. O eixo horizontal representa as épocas (normalizadas) e o eixo vertical o erro da RNA. A legenda exibe o número de camadas e o erro final atingido;
- **Gráfico da Variação da Saída**: exibe a variação da saída da rede ao longo das épocas, permitindo observar a estabilização da resposta durante o treinamento.

A.5 Aba *Gráfico da Saída da RNA*

A aba *Gráfico da Saída da RNA*, ilustrada na Figura 49, apresenta a resposta da rede treinada em relação aos padrões de entrada. Os dois gráficos exibidos são:

- **Gráfico da Saída**: mostra a saída atual da RNA para cada padrão de entrada do conjunto de dados, permitindo avaliar visualmente a qualidade do mapeamento aprendido. A barra lateral deslizante indica o índice da camada (ramo) cujos pesos estão sendo visualizados;

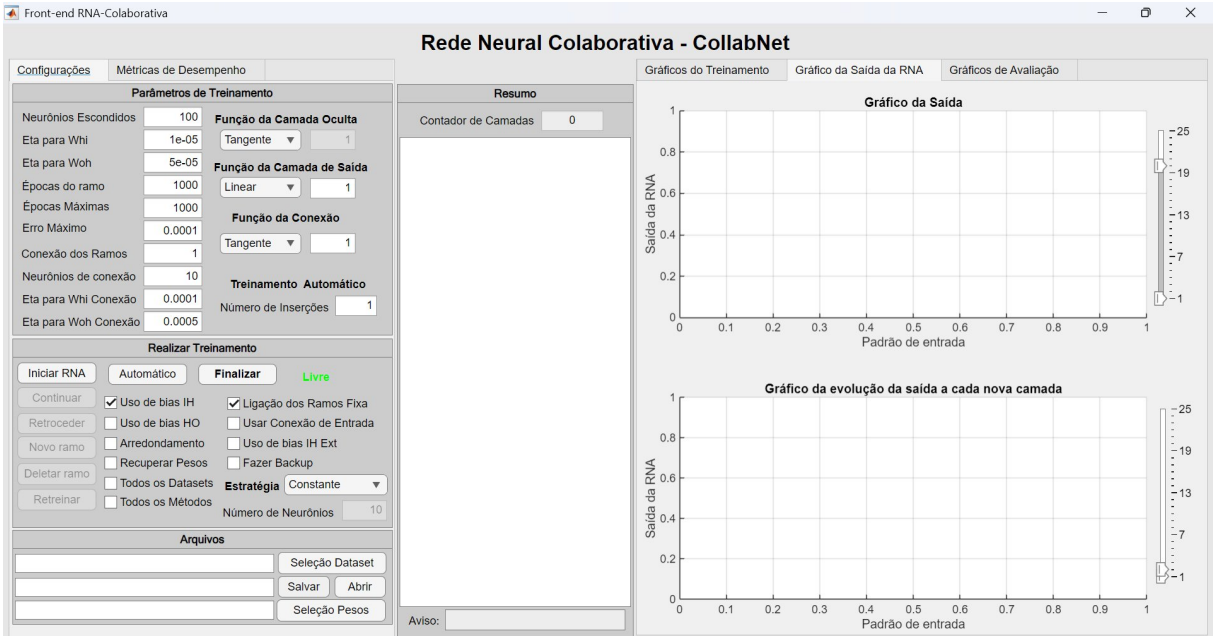


Figura 49 – Aba *Gráfico da Saída da RNA* da interface CollabNet.

- **Gráfico da Evolução da Saída a Cada Nova Camada**: apresenta como a saída da rede se transforma a cada novo ramo inserido, evidenciando a contribuição incremental de cada camada colaborativa para a melhoria da resposta.

A.6 Aba *Gráficos de Avaliação*

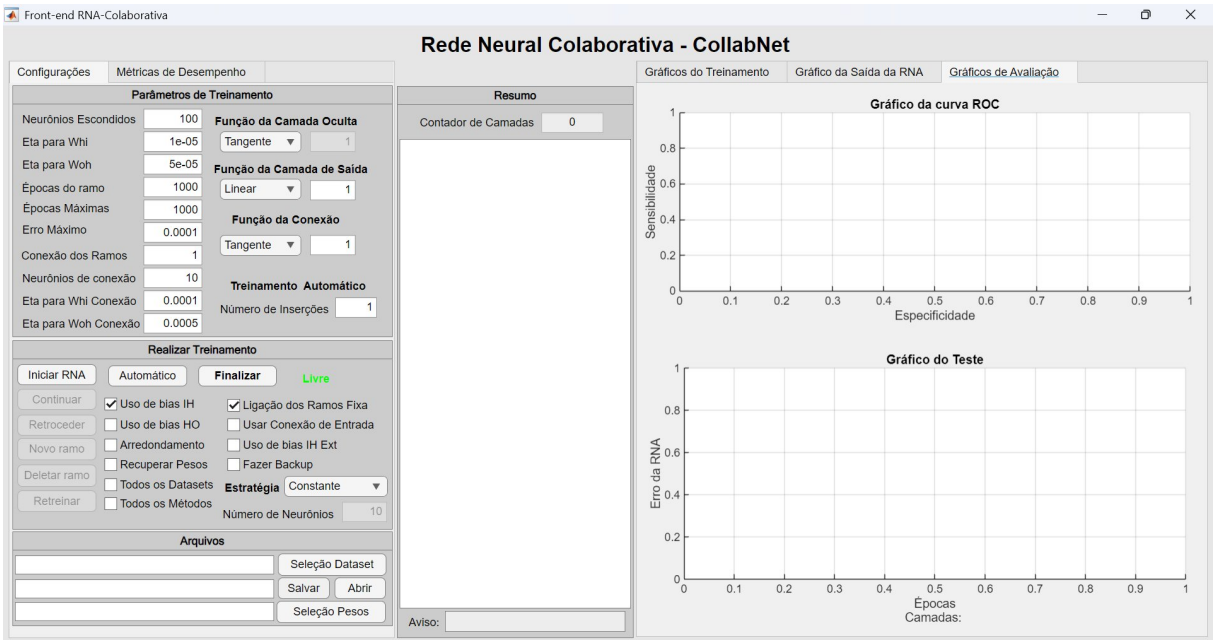


Figura 50 – Aba *Gráficos de Avaliação* da interface CollabNet.

A aba *Gráficos de Avaliação*, mostrada na Figura 50, reúne visualizações destinadas à análise do desempenho da rede no conjunto de teste:

- **Gráfico da Curva ROC** (*Receiver Operating Characteristic*): plota a sensibilidade em função da especificidade para diferentes limiares de classificação, sendo especialmente relevante para tarefas de classificação binária. A área sob a curva (AUC) é um indicador consolidado da capacidade discriminativa do modelo;
- **Gráfico do Teste**: exibe a evolução do erro da RNA sobre o conjunto de teste ao longo das épocas e camadas, de maneira análoga ao Gráfico do Erro da aba de treinamento, porém avaliado sobre dados não vistos durante o ajuste dos pesos.