UNIVERSIDADE FEDERAL DO MARANHÃO

Programa de Pós-Graduação em Ciência da Computação

Moisés Laurence de Freitas Lima Junior

Deep CollabNet: Rede Deep Learning Colaborativa

São Luís
2018

MOISÉS LAURENCE DE FREITAS LIMA JUNIOR

Deep CollabNet - Rede Deep Learning Colaborativa

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

UNIVERSIDADE FEDERAL DO MARANHÃO - UFMA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO - PPGCC

Orientador: Prof. Dr. Areolino de Almeida Neto

SÃO LUIS - MA

2018

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

Laurence de Freitas Lima Junior, Moisés.
Deep CollabNet - rede deep learning colaborativa / Moisés Laurence de Freitas Lima Junior. - 2018.
51 f.

Orientador(a): Areolino de Almeida Neto.
Dissertação (Mestrado) - Programa de Pós-graduação em Ciência da Computação/ccet, Universidade Federal do Maranhão, São Luis - MA, 2018.

1. Aprendizado profundo. 2. Deep feedforward. 3. Deep stacked autoencoder. I. de Almeida Neto, Areolino. II. Título.

MOISÉS LAURENCE DE FREITAS LIMA JUNIOR

Deep CollabNet - Rede Deep Learning Colaborativa

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Aprovada em 02 de maio de 2018

BANCA EXAMINADORA

Prof. Dr. Areolino de Almeida Neto (Orientador)
Universidade Federal do Maranhão - UFMA

Prof. Dr. Geraldo Braz Junior
Universidade Federal do Maranhão - UFMA

Prof. Dr. Sérgio Ronaldo Barros dos Santos
Universidade Federal de São Paulo - UNIFESP

Prof. Dr. Will Ribamar Mendes Almeida
Centro Universitário do Maranhão - UNICEUMA

SÃO LUIS - MA
2018

AGRADECIMENTOS

Primeiramente a Deus que iluminou o meu caminho durante essa caminhada.

Em seguida a minha esposa Kelinne e a nossa princesa Sofia, por terem sofrido e se alegrado comigo com cada batalha vencida nessa caminhada.

Em seguida aos meus pais e irmãs pelo amor, incentivo e apoio incondicional.

Ao meu orientador, pelo suporte durante todo o trabalho.

Aos amigos da república, pela amizade e companheirismo.

Aos colegas de laboratório pelo compartilhamento de idéias.

Ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão, pela oportunidade de desenvolver este trabalho.

RESUMO

Visando aprimorar o aprendizado de redes neurais profundas, neste trabalho é proposta a rede CollabNet, que consiste em um novo método de inserção de novas camadas escondidas em redes neurais do tipo *Deep FeedForward*, alterando o método tradicional de empilhamento de *autoencoders*. A nova forma de inserção é considerada colaborativa e busca a melhoria do treinamento em relação a abordagens baseadas em *autoencoders* empilhados. Nesta nova abordagem, a inserção de uma nova camada é realizada de maneira coordenada e gradual, mantendo sob controle do projetista a influência dessa nova camada no treinamento e não mais de modo aleatório e estocástico como no empilhamento tradicional. A colaboração proposta neste trabalho consiste em fazer com que o aprendizado da camada recém inserida continue o aprendizado obtido pelas camadas anteriores, sem prejuízo ao aprendizado global da rede. Desta forma, a camada recém inserida colabora com as camadas anteriores e o conjunto trabalha de forma mais alinhada ao aprendizado. A CollabNet foi testada na base de dados *Wisconsin Breast Cancer Dataset*, obtendo resultados satisfatórios e promissores.

Palavras-chave: Aprendizado profundo, Deep Feedforward, Deep Stacked Autoencoder.

ABSTRACT

In order to improve the learning of deep neural networks, this work presents the CollabNet network, a new method of insertion of new layers into a Deep FeedForward neural networks, changing the traditional stacked autoencoders method. This new way of insertion is considered collaborative and seeks to improve training against approaches based on stacked autoencoders. In this new approach, the insertion of a new layer is performed in a coordinated and gradual manner, keeping under designer's control the influence of the new layer on the training and no longer as random and stochastic as in traditional stacking. The collaboration proposed in this work consists of making the learning of the new inserted layer continues the learning obtained by the previous layers, without prejudice to the global learning of the network. In this way, the new inserted layer collaborates with the previous layers and the set of layers works in a way more aligned to the learning. CollabNet was tested in the Wisconsin Breast Cancer Dataset, obtaining satisfactory and promising results.

Keywords: Deep Learning, Deep Feedforward, Deep Stacked Autoencoder.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

Tahn	*Tangente Hiperbólica*
ReLU	*Rectified Linear Unit*
GELM-AE	*Generalized Extreme Learning Machine Autoencoder*
DCN	*Deep Convex Net*
DBN	*Deep Belief Network*
RNA	*Rede Neural Artificial*
DFF	*Deep Feedforward*
MLP	*Multilayer Perceptron*
BP	*Backpropagation*
CNN	*Convolutional Neural Network*
RBM	*Restricted Boltzmann Machines*
DBM	*Deep Boltzmann Machines*
GUI	*Guia de Interface de Usuário*
ROC	*Receiver Operating Characteristic*
CV	*Cross validation*
UCI	*Universidade da California Irvine*
RBM	*Restricted Boltzmann Machines*
DSA	*Deep Stacked Autoencoders*
EQM	*Erro Quadrático Médio*
TVP	*Taxa de Verdadeiros Positivos*
TFP	*Taxa de Falsos Positivos*

LISTA DE SÍMBOLOS

$f^{(n)}$	Camadas escondidas
$\hat{f}(x_i)$	Saída do neurônio x_i
y_i	Saída da camada escondida
net_n	Vetor com entrada de um neurônio
out_n	Vetor com saída de um neurônio
w_n	Peso da conexão entre os neurônios
e_n	Erro de saída de um neurônio
f_i'	Derivação da função de ativação
ε	Erro
D	Máscara de pesos
Whi	Pesos entre camadas
Y	Saída da rede
ΔD	Variação de máscara
c	Termo de atualização
ϕ	Função de ativação
ϕ_A	Função de ativação alterada
c	Parâmetro que define o comportamento do treinamento
eta	Taxa de aprendizado
U_i	Matriz de peso superior da DSA

SUMÁRIO

1 INTRODUÇÃO

As novas ferramentas de tecnologia da informação têm causado grande impacto na sociedade, sendo que o acesso ou não do indivíduo a essas tecnologias causam alterações estruturais e funcionais na sociedade em que se encontram inseridos. Nesse contexto, a tecnologia apresenta-se como uma importante ferramenta no intuito de facilitar e auxiliar as atividades do cotidiano humano. Sua presença pode ser percebida em praticamente tudo que fazemos, uma vez que muitas das nossas atividades diárias são realizadas com produtos tecnológicos. Diante desse cenário, técnicas de aprendizado de máquinas ganham cada vez mais espaço, considerando que essas técnicas visam à automatização das ações das máquinas de modo inteligente.

É cada vez mais frequente o uso de técnicas de aprendizado de máquinas nas mais variadas tarefas do dia-a-dia e o crescimento desta área de conhecimento segue em ritmo acelerado. Esse crescimento é decorrente de alguns fatores, entre eles estão: o crescente volume e variedade de dados disponíveis, o processamento computacional e o armazenamento de dados, ambos cada vez mais baratos. Assim, é possível citar alguns exemplos de sua aplicabilidade, como: a filtragem de conteúdo em redes sociais, recomendações de *sites*, identificação de objetos presentes em imagens e/ou vídeos, transcrição de voz em texto, diagnóstico de doenças, etc.

Embora os estudos das técnicas de aprendizado de máquinas terem seu inicio na década de 60, foi somente a partir da utilização de técnicas de *deep learning* que esta área de conhecimento começou a apresentar desempenho similar a humanos em problemas complexos e alguns casos até superando. O desempenho de algoritmos profundos pode ser atestado em diversas competições de aprendizado de máquinas espalhados pelo mundo a fora.

As possibilidades apresentadas em virtude da utilização de técnicas de *deep learning* fez com que a sub-área de aprendizado de máquinas, conhecida como *deep learning* (aprendizado profundo), tivesse um crescimento em ritmo acelerado. Tal fato pode ser comprovado observando o aumento expressivo no número de trabalhos acadêmicos abordando o tema e, em especial, os que visam à concepção de novos algoritmos e abordagens de treinamento, novas estruturas profundas, funções de ativação, novos métodos de regularização de dados, entre outras questões. Essas novas abordagens em sua grande maioria são derivações de abordagens conceituadas, tendo como o foco alguma melhoria em algum aspecto da abordagem de *deep learning* utilizada.

As abordagens mais difundidas e exploradas são as redes convolucionais, as redes profundas de crença, as máquinas de *boltzmann*, as redes generativas e *autoenconders*, sendo que esta última, serviu de principal inspiração para concepção da ideia implementada neste trabalho, que traz uma inovação no que tange à inserção e ao treinamento de novas camadas em redes totalmente conectadas.

1.1 JUSTIFICATIVA

Diversos trabalhos propondo novas arquiteturas de redes profundas vêm sendo apresentados a comunidade científica (GOODFELLOW et al., 2014). Essas novas arquiteturas visam à resolução de problemas nas diversas áreas do conhecimento humano. No entanto, grande parte desses trabalhos utilizam métodos estocásticos na inicialização e na inclusão de novas camadas escondidas. Além disso, o treinamento é feito de modo não-supervisionado e em alguns casos é realizado um treinamento híbrido, onde na primeira etapa do treinamento é utilizado aprendizado não-supervisionado com o objetivo de identificar as características presentes nos dados. Em seguida, o treinamento é realizado de forma supervisionada com intuito de classificar as características identificadas na etapa anterior.

A utilização dos métodos estocásticos retarda o aprendizado, pois em decorrência de sua aleatoriedade, a tendência natural é que haja uma perturbação no erro. Na tentativa de otimizar esse aprendizado e aumentar a profundidade de uma rede neural durante o treinamento, perturbando minimamente o erro da rede, surgiu o presente trabalho. Tendo como pretensão propor um método não estocástico de treinamento supervisionado para redes profundas, de modo que o projetista possua um maior controle sobre as novas camadas inseridas ao longo de todo o treinamento. Esse controle é dado ao projetista da rede por meio da regulação da influência da camada recém inserida no treinamento.

1.2 OBJETIVOS

O objetivo principal deste trabalho é apresentar uma proposta de inserção de novas camadas em uma rede neural do tipo *Deep Feedforward* (DFF), de forma que elas trabalhem colaborativamente no aprendizado da rede neural como um todo.

São objetivos específicos do trabalho:

- Apresentar um novo método de treinamento de rede profunda, utilizando o algoritmo do *backpropagation* para minimizar o erro do aprendizado.
- Realizar a integração colaborativa entre camadas escondidas, com intuito de promover a busca do valor mínimo do erro, mesmo em uma nova dimensão.
- Aplicar a abordagem proposta na base de dados de um problema real, no intuito de atestar a eficiência do método proposto.

1.3 TRABALHOS RELACIONADOS

Deep learning é uma área do aprendizado máquinas que está em constante evolução, desta forma, o número de novos algoritmos, estratégias e arquiteturas implementando esta técnica

é cada vez maior. Portanto esta seção é dedicada à apresentação de algumas abordagens que de alguma forma trouxeram inovação para a área e serviram de base conceitual ao presente trabalho.

Em Vincent e Larochelle (2010), é apresentada uma estratégia para construção de redes profundas com base em camadas de empilhamento de *autoencoders denoising*, que são treinados individualmente para restaurar as versões corrompidas de suas entradas. Essa abordagem é uma variante do *autoencoder* tradicional, diferenciando-se na etapa de reconstrução da entrada. Enquanto a abordagem tradicional possui basicamente duas etapas: o codificador, que transforma um vetor de entrada x em uma representação oculta y e o decodificador, que a partir da representação oculta y mapeia de volta para um vetor z reconstruído no espaço de estados. Nesse trabalho, existe a adição de um elemento chamado *denoising autoencoder*, que é treinado para reconstruir uma entrada "reparada" de uma versão corrompida do vetor x, por meio de um mapeamento estocástico da entrada $\tilde{x} \sim q_D(\tilde{x}|x)$.

No trabalho de Netanyahu (2016), é proposta uma nova função de ativação, que implementa mapeamentos ortogonais não-lineares baseados em permutações usando *autoencoders* convolucionais profundos. A OPLU, assim nomeada, foi testada em redes *feedforward* e recorrentes, apresentando um desempenho semelhante a outras funções de ativação já conceituadas, como a Tanh e a ReLU. A função de ativação OPLU tem como característica básica preservar a norma dos gradientes oriundos do *backprogation*.

Em Badrinarayanan, Kendall e Cipolla (2017), é apresentada uma nova arquitetura de rede neural profunda convolutiva para segmentação semântica *pixel-wise*, denominada SegNet. Este mecanismo de segmentação consiste em uma rede de codificação, uma outra de decodificação e uma camada de classificação de *pixels*. O grande diferencial dessa abordagem está na implementação da rede de decodificação, a qual usa índices de agrupamento (*pooling*) calculados na etapa de *max-pooling* do codificador, para realizar um aumento artificial na quantidade de amostras de seus mapas de entrada. A rede SegNet é uma arquitetura eficiente para compreensão de cenas em movimento, tanto em relação a quantidade de memória utilizada, quanto em relação ao tempo de treinamento.

Em Yu e Deng (2011), foi desenvolvida uma arquitetura profunda escalável voltada para a classificação de padrões de voz, denominada *Deep Convex Net* (DCN). A arquitetura da rede convexa profunda foi projetada para solucionar o problema da escalabilidade típica do grande vocabulário de palavras presente no problema do reconhecimento de voz. O aprendizado da DCN é baseado em uma estrutura de lotes, em vez de estocástica, o que traz ao algoritmo a capacidade de ser executado paralelamente em várias máquinas. Este algoritmo foi testado nas tarefas MNIST e TIMIT, demonstrando um desempenho superior ao ser comparado às *Deep Belief Network* (DBN). Neste trabalho é ainda destacado que a superioridade desta abordagem não se refere apenas a alta escalabilidade e distribuição de treinamento, mas também na precisão da classificação em ambas as tarefas.

Em Sun et al. (2017), é proposto um algoritmo de treinamento de redes neurais profundas

baseado em máquina de aprendizagem extrema, chamado de *Generalized extreme learning machine autoencoder* (GELM-AE). Essa nova variante combina a máquina de aprendizagem extrema com *autoencoder*, utilizando uma espécie de regularização múltipla. O GELM-AE foi testado em conjuntos de dados de problemas reais, seus resultados superam alguns algoritmos clássicos de aprendizado não supervisionado, tais como o *k-means* (MACQUEEN, 1967), o *laplacian embedding* (WANG; HUANG; MAKEDON, 2014), o *spectral clustering* (XU; WUNSCH II, 2005) e o próprio *extreme machine learning autoencoder* (HUANG; ZHU; SIEW, 2006). Ainda em Sun et al. (2017), foi proposta também a rede neural profunda denominada *multilayer generalized extreme learning machine autoencoder*, que por sua vez empilha vários GELM-AE para detectar representações mais abstratas.

Em Makhzani e Frey (2013), foi proposto o *autoencoder k-sparse*, que é um autocodificador com função de ativação linear, onde nas camadas ocultas somente as atividades k mais altas são mantidas. Este algoritmo é destacado pela sua facilidade de treinamento e codificação, o que o torna bem adaptado a problemas com grande quantidade de dados.

Em Romero et al. (2014), é proposta uma nova estrutura para comprimir redes largas e profundas em redes finas com uma profundidade maior. Nessa abordagem, existe a figura do professor para guiar o processo de aprendizado do aluno, de modo que a rede estudante (FitNet) aprenda uma representação intermediária que seja preditiva das representações intermediárias da rede de professor. De acordo com Romero et al. (2014), a FitNet é adequada para aplicações com limitações de memória ou de tempo, haja vista que essa abordagem permite a formação de redes mais finas e mais profundas. Os resultados apresentados nesse trabalho confirmam que modelos mais profundos generalizam melhor e ao passo que esses modelos são tornados mais finos, eles reduzem o custo computacional de forma significativa.

Em Bengio et al. (2006), é estudado o algoritmo do *greedy layer-wise*, que foi inicialmente proposto por Hinton e Salakhutdinov (2006). Nesse trabalho é realizado um estudo minucioso da aplicação do algoritmo em redes profundas, apresentando novas maneiras de lidar naturalmente com entradas de valores contínuos, de forma que as camadas superiores representam abstrações relevantes de alto nível, enquanto as camadas mais profundas representam especificações de granularidade mais finas. Este algoritmo foi aplicado com sucesso inicialmente nas redes generativas *Deep Belief Networks* e depois expandida a outros tipos de redes profundas.

1.4 ORGANIZAÇÃO DO TRABALHO

Este capítulo introduziu o tema da pesquisa, destacando seus objetivos e descrevendo um conjunto de trabalhos que propõe novas arquiteturas de redes *deep learning*e métodos de treinamento.

O capítulo seguinte apresenta os principais conceitos a respeito da sub-área de conhecimento de aprendizado de máquinas, *Deep learning*. É apresentado nesse capítulo também um

breve histórico de *deep learning* e ainda uma taxonomia básica dos tipos de abordagens mais utilizados.

O Capítulo 3 apresenta a metologia proposta, relatando detalhes da estrutura e do treinamento da rede CollabNet. Nesse capítulo também é apresentado em detalhes a alteração realizada na função de ativação sigmóide, denominada $sigmoid_A$.

Em seguida, é apresentado o Capítulo 4 contendo relatos das experiências da aplicação da metodologia proposta no reconhecimento de padrões na base *Wisconsin Breast Cancer Dataset* (WOLBERG; STREET; MANGASARIAN, 2011). A partir dos referidos testes, são apresentados os resultados que atestam a eficiência do método, demonstrando que nele possui um futuro promissor na resolução de problemas de classificação e reconhecimento de padrões.

Baseado nos resultados obtidos, o último capítulo mostra as conclusões observadas, bem como contribuições obtidas. Por fim, são apontadas sugestões de pesquisas que podem dar prosseguimento a este trabalho.

2 *DEEP LEARNING*

2.1 UM BREVE HISTÓRICO DE *DEEP LEARNING*

O início dos estudos em redes neurais artificiais (RNA) remonta aos anos 1940, a partir dos estudos de McCulloch e Pitts. Os trabalhos dessa época consistiam basicamente de uma variação de uma regressão linear, sendo que tais redes tinham uma profundidade máxima de uma camada. Essa abordagem de RNA seguiu por várias décadas (HAYKIN, 1999). Já os primeiros modelos com várias camadas sucessivas de neurônios não lineares surgiram em meados dos anos de 1960, principalmente apoiados pelo algoritmo de treinamento *Group Method of Data Handling*, sendo que estes podem ser considerados como os primeiros sistemas *deep learning* (SCHMIDHUBER, 2014).

Ainda nos anos de 1960 e 1970, foi proposto um método de minimização de erros através da decida do gradiente. Esse método, baseado na aprendizagem supervisionada propõe uma eficiente decida do gradiente do erro de saída da rede pelo gradiente deste erro em função dos pesos. Sua aplicação em redes com profundidades diversas ocorreu apenas na década de 1980, sendo batizado de *Backward-Error-Propagation*, depois simplificado para *backprogation* ou simplesmente BP. No entanto, após alguns anos de pesquisas e inúmeros experimentos com redes neurais treinadas com *backprogation*, foi possível constatar que, ao contrário do que se imaginava no início dos anos 80, o BP não seria uma solução para todos os problemas de aprendizado de máquinas. De um modo geral, embora o *backprogation* possibilitasse treinamento em problemas profundos, os resultados iniciais apontavam que este algoritmo trabalhava bem apenas com redes rasas, o que tornou o uso do BP em redes profundas quase que inviáveis no final de década de 1980 (SCHMIDHUBER, 2014).

Desta maneira, pesquisadores de redes neurais só tiveram maior interesse em técnicas de *deep learning* em meados dos anos 2000. Tal interesse pode ser atribuído ao fato de que, nessa época, o desempenho dos algoritmos de *deep learning* começou a apresentar resultado similar ou por vezes superior em relação à outros métodos de aprendizado de máquinas em diversos problemas importantes (GOODFELLOW; BENGIO; COURVILLE, 2016). No ano de 2009, algoritmos de *deep learning* ganharam várias competições oficiais de reconhecimento de padrões, atingindo resultados semelhantes a humanos no reconhecimento de determinados padrões, tornando-se assim, uma nova área de pesquisa de grande relevância para o aprendizado de máquinas no geral (SCHMIDHUBER, 2014). Técnicas de aprendizado profundo, além de bater recordes em diversos problemas de aprendizado de máquinas, têm sido aplicadas com sucesso em diversas atividades distintas, como prever atividades moleculares, análise de dados, reconstrução de circuitos cerebrais, prever mutações no código genético, etc. (LECUN; BENGIO; HINTON, 2015).

2.2 CONCEITOS DE *DEEP LEARNING*

Baseado nos conceitos de RNA e inspirado em estudos biológicos do cérebro humano e no córtex visual dos mamíferos, surge o *deep learning*. Estes são algoritmos que, a grosso modo, realizam a inclusão de diversas camadas escondidas em uma rede neural artificial na resolução de problemas cada vez mais complexos (LAROCHELLE et al., 2009).

Os algoritmos que implementam *deep learning* buscam, de um modo geral, a identificação de abstrações dos dados, partindo da identificação dos níveis mais baixos e chegando nos níveis mais altos, de forma que, por meio da composição das características de nível mais baixo, obtenham-se as características de nível mais alto e, consequentemente, novas representações. Desta forma, o aprendizado de características em múltiplos níveis de abstração permite ao sistema computacional aprender complexas funções de mapeamento dos dados de entrada para saída, de maneira independente de características criadas manualmente. Ou seja, esta técnica pode ser considerada uma forma de automatizar a geração de características que sejam mais representativas de um determinado problema de reconhecimento de padrões (BENGIO, 2009; LECUN; BENGIO; HINTON, 2015).

Deep learning é uma nova subárea de aprendizado de máquinas, que possui grande variedade de técnicas e de arquiteturas de aprendizado, com uso de muitas camadas de processamento não-linear de informações de natureza hierárquica, para extração de características e análise de padrões. As redes *deep learning* podem utilizar algoritmos de aprendizagem supervisionadas ou não-supervisionada. Outra característica básica do aprendizado profundo é o uso frequente de redes neurais artificiais como base dos seus algoritmos. No entanto, sua aplicação não se dá apenas em RNA, tendo abordagens baseadas em máquinas de vetores de suporte, entre outras (DENG; YU, 2013).

Diante do vasto número de algoritmos e técnicas de aprendizado profundo, é possível categorizá-las em três grandes classes, de acordo com a forma de treinamento. São elas:

- Redes profundas para aprendizado não supervisionado
- Redes profundas com aprendizado supervisionado
- Redes híbridas

Aprendizado não supervisionado

As redes profundas com aprendizado não supervisionado buscam realizar a correlação em alto nível dos dados para fins de classificação de padrões, quando não há informações sobre rótulos de classes disponíveis. Ou seja, refere-se a não usar supervisão de tarefas no processo de aprendizagem, com rótulos de classes, por exemplo. Referem-se também às técnicas que buscam caracterizar distribuições estatísticas dos dados, em relação a associação de classes. Muitas redes

profundas nesta categoria podem ser usadas para gerar amostras de maneira significativa através da amostragem das redes, com exemplos de *Restricted Boltzmann Machines* (RBM), *Deep Belief Network* (DBN), *Deep Boltzmann Machines* (DBM) e *Denoising Autoencoders* generalizados e, portanto, são considerados modelos generativos.

Aprendizado supervisionado

As redes profundas com aprendizado supervisionado trabalham com a ideia de discriminação de dados para fins de classificação de padrões, onde se classificam os novos dados a classes condicionadas aos dados previamente apresentados. Os dados do rótulo alvo estão sempre disponíveis em formas diretas ou indiretas para essa aprendizagem supervisionada.

Híbridas

Nos modelos híbridos, o objetivo é obter uma máxima discriminação dos dados, obtendo isso com uma melhor otimização e/ou regularização das redes profundas supervisionadas, utilizando redes não supervisionadas para estimar os parâmetros, bem como classificar os dados genericamente.

2.3 ABORDAGENS *DEEP LEARNING*

Diante das classificações das redes de aprendizado profundo, citadas anteriormente, é possível apresentar uma vasta gama de técnicas que implementam de alguma maneira uma rede neural profunda. Nesta seção são apresentadas técnicas que motivaram e/ou serviram de base para o presente trabalho.

2.3.1 Redes neurais *deep feedforward*

As redes *deep feedforward* (DFF) são consideradas uma extensão das redes *Multilayer Perceptron* (MLP). As redes DFF apresentam-se como um importante *framework* de aprendizado profundo, sendo a essência dos modelos de *deep learning* (GOODFELLOW; BENGIO; COURVILLE, 2016). O modelo *deep feedforward* caracteriza-se pela adição de diversas camadas escondidas à rede, de modo que seja possível representar funções com complexidade cada vez maior.

As redes *feedforward* possuem seus conceitos baseados em alguns conceitos da neurociência. Sendo que este tipo de rede tem como característica básica o fato de trabalharem com muitas camadas sequenciais de neurônios, em conjunto com funções $f^{(i)}(x)$ utilizadas para calcular as representações dos vetores-valores. Este modo de trabalho assemelha-se ao funcionamento do cérebro biológico (BABRI; TONG, 1996). No entanto, a modelagem perfeita do cérebro não é o foco das RNAs. Pesquisas modernas em RNA são baseadas em disciplinas de matemática e engenharia, sendo que essas se propõem alcançar a generalização estatística por

meio de funções de aproximação (GOODFELLOW; BENGIO; COURVILLE, 2016; HAYKIN, 1999).

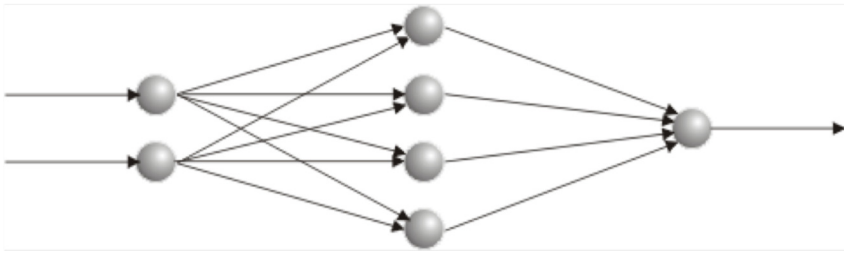
No ressurgimento dos estudos em aprendizado profundo, em meados do ano de 2006, as redes *deep feedforward* tiveram uma má reputação, pois testes empíricos atestavam que estas redes não tinham um bom desempenho com muitas camadas escondidas se não tivessem um auxílio de outros modelos, como os probabilísticos. Hoje, é sabido que, com recursos computacionais e práticas corretas de construção, as redes *deep feedforward* possuem um bom desempenho e a aprendizagem baseada em gradientes com redes *feedforward*, que até meados do ano de 2012 era vista como uma solução exclusiva para redes rasas, passa então a ser largamente utilizada no desenvolvimento de modelos probabilísticos, como as variações do *autoencoder* e as redes generativas adversáriais (GOODFELLOW et al., 2014; GOODFELLOW; BENGIO; COURVILLE, 2016).

A estratégia apresentada neste trabalho visa à concepção de uma rede *deep feedforward* colaborativa, onde a partir dos conceitos e arquiteturas de diversas abordagens profundas, tem-se o modelo CollabNet. As próximas seções são dedicadas a fazer um breve relato acerca destas metodologias que apoiaram o desenvolvimento desta pesquisa.

Redes Multilayer Perceptron

Redes Neurais do tipo MLP possuem sua estrutura em camadas de neurônios artificiais, sendo que tais camadas estão dispostas de acordo com a arquitetura estritamente *feedforward*, conforme Figura 1. Com esta arquitetura, o fluxo de dados segue da camada de entrada até a saída, passando obrigatoriamente pela camada escondida (intermediária).

Figura 1 – Estrutura estritamente *feedforward*



Fonte: de Almeida Neto (2013)

A rede MLP é uma rede totalmente conectada, ou seja, todos os neurônios são interconetados. A entrada de um neurônio é a soma da saída dos neurônios conectados a ele, multiplicada pelos respectivos pesos das conexões (Equação 2.1). A saída dos neurônios é dada por meio do cálculo da função de ativação presente em cada neurônio, sendo o cálculo desta função realizado

a partir do somatório de todas as suas entradas (HAYKIN, 1999).

$$net_h = \sum_{i=1}^{n} out_i \cdot w_{hi} \tag{2.1}$$

onde net_h é chamada de entrada líquida do neurônio da camada escondida e é igual ao somatório de suas entradas, i define uma unidade da camada anterior, out_i é a saída do neurônio i e w_{hi} é o peso da conexão entre os neurônios h e i.

As camadas intermediárias realizam o processamento a partir da saída da camada anterior e calculam sua saída que é levada à camada seguinte, conforme a Equação 2.2:

$$out_h = \phi^h(net_h) \tag{2.2}$$

onde ϕ^h é a função de ativação aplicada ao neurônio h da camada escondida, geralmente não linear, sendo as funções sigmóide e tangente hiperbólica as mais comuns.

A camada de saída produz a saída final da rede, que é utilizada para calcular o erro da rede em relação à saída desejada. Os dados de entrada e saída são obtidos de modo análogo ao das camadas intermédiarias, conforme as Equações 2.1 e 2.2, respectivamente. No entanto, a função de ativação geralmente é linear. O erro produzido pela rede é utilizado posteriormente para ajustar os pesos das conexões intercamadas.

Existem diversos algoritmos focados em realizar o ajuste dos pesos de uma rede MLP, sendo o Levenberg-Marquart, regularização bayesiana e o *backprogation* os mais difundidos (GAMA et al., 2015). O algoritmo *backprogation* foi adotado nesta pesquisa, portanto, o mesmo será tratado na subseção seguinte.

Algoritmo *backpropation*

O algoritmo *backprogation* (RUMELHART; HINTON; WILLIAMS, 1986) é largamente utilizado no treinamento de redes neurais multicamadas, este algoritmo é baseado no gradiente descendente e para que seja possível sua utilização, necessariamente a função de ativação precisa ser contínua, diferenciável e, de preferência, não decrescente (GAMA et al., 2015). Nesse projeto a função de ativação utilizada foi a sigmóide, que em seções posteriores é relatado os motivos de sua escolha.

O *backprogation* é constituído de duas fases distintas, uma para frente (*forward*) e uma pra trás (*backward*). A fase *forward* é explicada na seção anterior, já a segunda fase é iniciada a partir da diferença entre os valores de saída produzidos pela rede e os valores desejados para cada neurônio e sua função é ajustar os pesos das conexões entre as camadas, iniciando pela camada saída até a primeira camada intermediária. A Equação 2.3 mostra o cálculo do erro retropropagado para a camada anterior:

$$e_j = \sum_{i=1}^{n} e_i \cdot f_i' \cdot w_{ij} \tag{2.3}$$

onde e_i representa o erro de saída do neurônio i, f_i' corresponde à derivada da função de ativação do neurônio i e w_{ij} é o peso entre o neurônio i da camada de saída e o neurônio j da camada escondida.

A Equação 2.4 apresenta o cálculo dos novos pesos entre as camadas de saída e escondida, nele os pesos são ajustados de modo que a cada iteração do algoritmo de treinamento o erro diminua através de uma pequena variação no valor de cada peso.

$$\Delta W_{ij} = \eta \cdot e_i \cdot f_i' \cdot out_j \tag{2.4}$$

onde ΔW_{ij} representa a variação dos pesos entre as camadas, η a taxa de aprendizado, e_i o erro de saída do neurônio i, f_i' é a derivada da função de ativação do neurônio i com relação à sua entrada net e out_j representa o valor de saída do neurônio j da camada escondida.

O Algoritmo 1 ilustra os principais passos do *backprogation*.

Algoritmo 1: Algoritmo de treinamento *backpropagation*

Entrada: Um conjunto de n objetos de treinamento
Saída: Rede MLP com valores dos pesos ajustados

```
início
    Inicializar pesos rede com valores aleatórios
    Inicializar erro_total = 0
    repita
        para cada objeto x_i do conjunto de treinamento faça
            para cada camada da rede, a partir da primeira camada intermediária faça
                para cada neurônio n_jl da camada atual faça
                    Calcular valor da saída produzida pelo neurônio, f̂
                fim
            fim
            Calcular erro_parcial = y − f̂
            para cada camada da rede, a partir da camada de saída faça
                para cada neurônio n_jl da camada atual faça
                    Ajustar pesos do neurônio utilizando a Equação 2.3
                fim
            fim
            Calcular erro_total = erro_total + erro_parcial
        fim
    até erro_total < ε;
fim
```

Fonte: Autor

Para que o algoritmo *backprogation* tenha uma solução satisfatória, é necessário que se observe alguns pontos importantes em sua execução. O ajuste de parâmetros é um destes pontos,

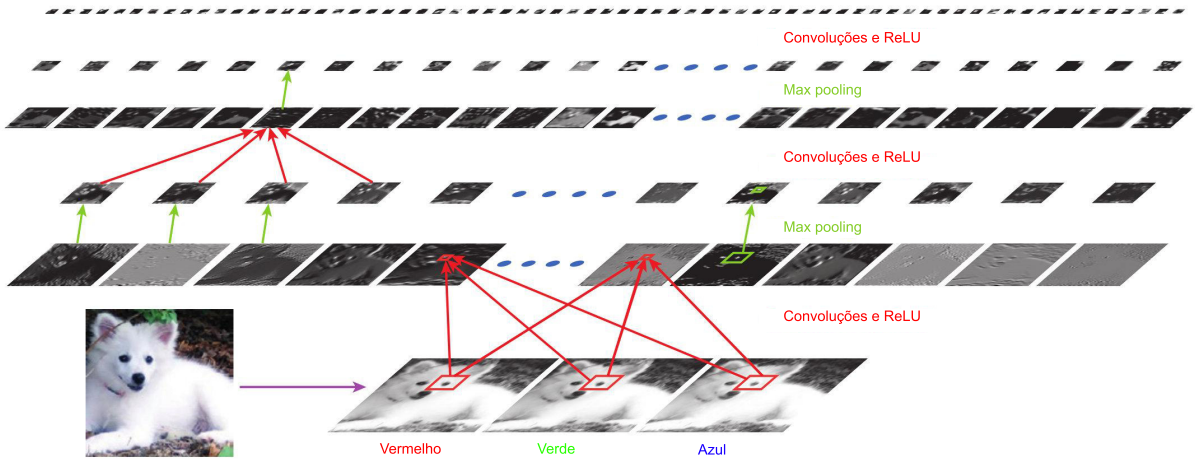
sendo o valor da taxa de aprendizagem um dos parâmetros mais importantes, pois este tem uma forte influência no tempo necessário à convergência da rede. Critérios de parada também são pontos de grande importância para que se evite esforços desnecessários (HAYKIN, 1999).

2.3.2 Redes Neurais Convolutivas

As redes neurais convolucionais (CNN - do inglês *convolutional neural network*) são consideradas um importante método de aprendizado profundo, especialmente projetado para lidar com a variabilidade em dados bidimensionais (2D), como as imagens em formato matricial. No entanto, as CNNs são versáteis e têm a capacidade de processar em diversos formatos de matrizes: 1D para sinais e sequências (linguagem); 2D para imagens ou espectrogramas de áudio; e 3D para imagens em vídeos ou volumétricas. As CNNs ou ConvNets possuem quatro princípios que a aproximam de sinais naturais: conexões locais, pesos compartilhados, agrupamento e o uso de muitas camadas (ALMOUSLI, 2014; LECUN; BENGIO; HINTON, 2015; PERONA; FINK, 2013).

A CNN é estruturada como uma série de estágios, sendo que os primeiros estágios são compostos por dois tipos de camadas: camadas convolucionais e de agrupamento (*pooling layer*). As unidades em uma camada convolucional são organizadas em mapas de características, onde cada unidade é conectada a um *patch* local nos mapas de recursos da camada anterior através de um conjunto de pesos. O resultado dessa soma ponderada passa então por uma função de ativação não-linear, chamada de unidade linear retificada (ReLU - sigla em inglês) . Na Figura 2, é apresentada a estrutura genérica de uma ConvNet.

Figura 2 – Estrutura Genérica da CNN



Fonte: Lecun, Bengio e Hinton (2015)

A arquitetura de CNN busca por algum grau de invariância a deslocamento, escala e distorção combinando as seguintes ideias arquiteturais (LECUN; BENGIO; HINTON, 2015):

- Tem seu mapeamento de características, ou campo receptivos locais, como um dos pontos de maior similaridade com as redes neurais biológicas.
- Pesos compartilhados permitem uma redução dos parâmetros livres e invariância geométrica.
- Sub-amostragem temporal ou espacial reduz o tamanho total dos mapas de características a cada camada, chegando assim na última camada apenas como valores unidimensionais, o que nesse ponto em diante as torna equivalentes a uma rede neural MLP.

Essas premissas da arquitetura da CNN possuem duas razões. Primeiro, em dados matriciais, como imagens, grupos de valores locais são comumente bastante correlacionados, formando tópicos ou características que são facilmente detectadas. Em segundo lugar, as estatísticas locais de imagens e/ou outros sinais são invariantes para a localização. Em outras palavras, se uma característica aparecer em uma determinada localização da imagem, tal padrão pode aparecer em qualquer outro lugar. Por este motivo, a CNN compartilha os mesmos pesos em locais diferentes da matriz da imagem (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.3.3 *Autoencoders*

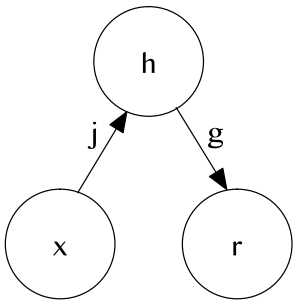
A utilização de *autoencoders* em redes neurais é comum a décadas, sendo utilizados principalmente para redução da dimensionalidade e compressão. Com grande capacidade de classificação e pré-formação, um *autoencoder* é, portanto, uma especificação de uma RNA, sendo considerado por muitos autores um caso especial das redes *feedforward*, cujos vetores de saída têm a mesma dimensionalidade que os vetores de entrada. O *autoencoder* é frequentemente usado para aprender uma representação ou uma codificação efetiva dos dados originais, sob a forma de vetores de entrada, em camadas ocultas. O *autoencoder* é um método de extração de recurso não-linear que não utiliza classes rotuladas. Como tal, os recursos extraídos pretendem preservar e representar melhor a informação, em vez de realizar tarefas de classificação, embora às vezes esses dois objetivos estejam correlacionados (GOODFELLOW; BENGIO; COURVILLE, 2016; DENG; YU, 2013).

Um *autoencoder* é muitas vezes treinado usando uma variação do BP e geralmente utilizando o método de descida estocástica do gradiente. No entanto, o BP não apresenta um bom desempenho em alguns casos onde existe uma grande quantidade de camadas ocultas. Uma vez que os erros propagam-se de volta para as primeiras camadas, eles tornam-se ínfimos e o treinamento ineficaz. Este problema é contornado em parte com algumas variações do *backprogation*. No entanto, ainda resultam em uma lenta aprendizagem e com soluções precárias, especialmente com quantidades limitadas de dados de treinamento (DENG; YU, 2013).

Um *autoencoder* é treinado de forma que a partir de uma entrada X, esta entrada é codificada em uma representação $c(X)$, de modo que a entrada pode ser reconstruída a partir

de $c(X)$. Consequentemente, a saída desejada de um *autoencoder* é a sua própria entrada (BENGIO, 2009). Basicamente, um *autoencoder* é uma RNA treinada para tentar copiar a sua entrada na saída. Internamente, possui uma camada oculta h que descreve um código utilizado para representar a entrada. A rede *autoencoder* consiste em duas partes: uma função codificadora $h = f(x)$ e um decodificador, que é responsável pela reconstrução $r = g(x)$, conforme a Figura 3. Caso o *autoencoder* consiga simplesmente definir $g(f(x)) = X$ para todas as entradas, então este *autoencoder* não está sendo útil, pois está apenas reproduzindo as entradas. Em vez disso, *autoencoders* devem ser projetados para não realizar uma cópia perfeita das entradas, normalmente, eles são restritos de modo a permitir copiar apenas aproximadamente e para copiar apenas a entrada que assemelha-se aos dados de treinamento (DENG; YU, 2013; KRIZHEVSKY; HINTON, 2011).

Figura 3 – Autoencoder



Fonte: Goodfellow, Bengio e Courville (2016)

Devido a facilidade de treinamento, os *autoencoders* têm sido utilizados como blocos de construção para formar uma rede profunda, onde cada nível está associado com um *autoencoder* que pode ser formado separadamente (GOODFELLOW; BENGIO; COURVILLE, 2016).

Deep Stacked Autoencoders

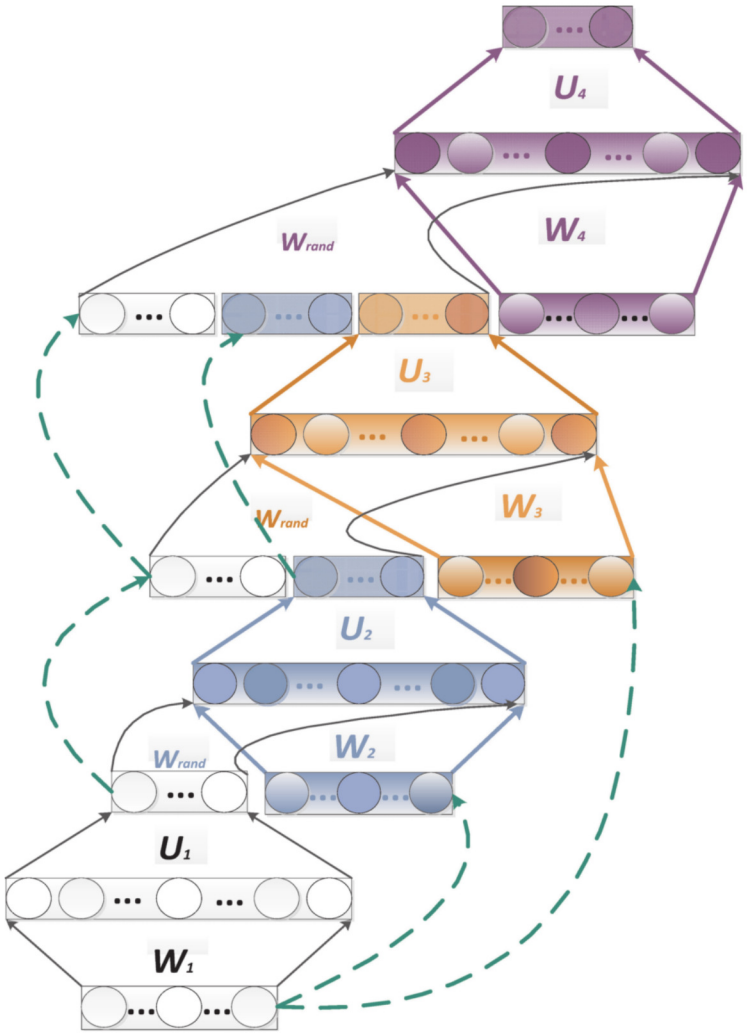
As *deep learning* apresentadas até este ponto são poderosos *frameworks* para tarefas de classificação e reconhecimento de diversos tipos de estruturas de dados. No entanto, tais técnicas não são triviais quando há a necessidade de paralelizar o aprendizado para obter-se um aprendizado em grande escala. No intuito de facilitar o aprendizado de máquina em grande escala por meio de algoritmos profundos, surgem as *Deep Stacked Autoencoders* (DSA), propostas inicialmente por Yu e Deng (2011) e voltadas para o reconhecimento de voz.

A técnica do DSA busca empilhar módulos simples de funções ou classificadores, de modo que estes módulos simples aprendam funções mais complexas. O DSA utiliza técnicas de aprendizado supervisionado para empilhar os módulos básicos, tal qual a rede MLP. Têm-se ainda funções sigmóidais não-lineares em camadas escondidas e saída linear em unidades de saída. A linearidade nas unidades de saída permite uma estimativa altamente eficiente, paralelizada e de forma fechada para os pesos da rede saída. Este tipo de arquitetura profunda foi

desenvolvido com estados escondidos adicionados para aplicações bem-sucedidas de linguagem natural e reconhecimento de voz, onde a informação de segmentação é desconhecida nos dados de treinamento. As redes neurais convolutivas também podem ser consideradas como uma arquitetura de empilhamento, mas as informações de supervisão normalmente não são usadas até no módulo de empilhamento final (DENG; YU, 2013).

A arquitetura básica de uma DSA possui um número variável de módulos em camadas, sendo que cada módulo é uma RNA especializada de apenas uma camada oculta e dois conjuntos de pesos treináveis. Essa estrutura é apresentada na Figura 4, onde são ilustrados quatro módulos, cada um apresentado com uma cor distinta e as linhas tracejadas indicam que a camada é uma cópia. É notório que em problemas reais é possível que uma DSA possua algumas centenas de módulos para resolução de certos problemas, como classificação de imagem e fala (DENG; YU, 2013; TUR et al., 2012).

Figura 4 – Arquitetura de uma DSA com quatro módulos empilhados



Fonte: Deng e Yu (2013), Tur et al. (2012)

A matriz de peso da camada inferior é denotada por W_i e conecta a camada de entrada linear à camada oculta não-linear. A matriz de peso superior é denotada por U_i e conecta a camada escondida não linear à saída linear. As conexões são em série, sobrepondo os módulos que possuem a mesma arquitetura - uma camada de entrada linear, seguida por uma camada oculta não-linear e finalizada com a camada de saída linear.

A abordagem do DSA serviu de inspiração para "criação" deste trabalho, ou seja, a ideia de empilhar várias camadas com módulos simples para obter uma arquitetura profunda supervisionada. A CollabNet, assim como a DSA, trabalha de forma que as unidades de saída de um módulo inferior são um subconjunto de unidades de entrada de um módulo adjacente. No entanto, novos módulos (camadas) em uma DSA recebem a saída de um módulo anterior sem nenhum tratamento, diferentemente da CollabNet que realiza um tratamento nos dados recebidos da camada anterior.

Os conceitos de DSA foram utilizados como inspiração para diversos outros trabalhos, sendo grande o número de novas arquiteturas que são inspiradas em DSA para obter um método de aprendizagem de máquinas eficiente para uma determinada área.

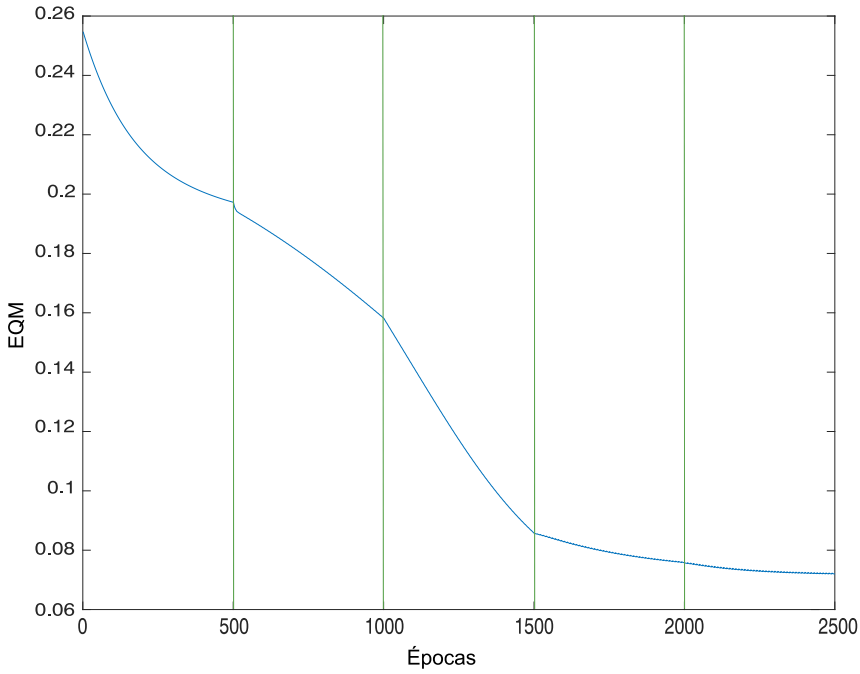
3 COLLABNET - *DEEP FEEDFORWARD* COLABORATIVA

O presente trabalho tem como foco a implementação de uma nova estratégia de inserção de camadas escondidas em uma rede *deep feedforward*, visando a evitar as instabilidades no erro de saída, durante o treinamento, provenientes de uma nova camada inserida pelos métodos tradicionais. A integração entre as novas camadas deve ser cuidadosamente observada para que o erro de saída da rede sempre convirja. Essa contínua convergência deve ser de tal forma, que uma nova camada sempre melhore o resultado das camadas anteriores e com isso acaba sempre havendo uma colaboração da nova camada.

Desta forma, busca-se neste trabalho uma maneira eficiente de aumentar a profundidade da rede, aumentando o número de camadas escondidas em tempo de execução do *software* de forma colaborativa.

Esse método parte do princípio que cada nova camada deve iniciar seu treinamento exatamente do ponto em que a camada imediatamente anterior parou. Em outras palavras, busca-se a diminuição do erro paulatinamente, mesmo quando outra camada é inserida na rede neural, conforme apresentado na Figura 5. Assim, esse esquema pode proporcionar aprendizado crescente, evitando armadilhas de mínimos locais ou platôs.

Figura 5 – Comportamento esperado do erro



Fonte: Autor

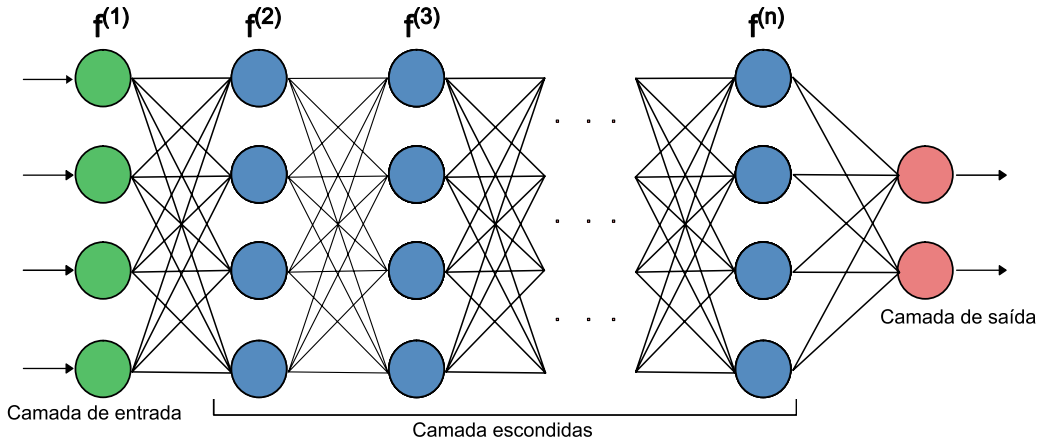
No entanto, para ter sucesso nessa tarefa, a integração da nova camada deve ser adequadamente realizada, caso contrário a nova camada pode afetar negativamente a aprendizagem já alcançada, provocando uma piora no aprendizado. Portanto, a ideia principal desta proposta é

desenvolver uma técnica para incorporar o aprendizado das camadas anteriores no treinamento da nova camada, a fim de fornecer conhecimento sobre o aprendizado já obtido para uma nova camada.

3.1 ESTRUTURA

Esta proposta consiste em adicionar camadas escondidas $f^{(n)}$ à estrutura inicial. A adição de novas camadas ocorre gradualmente, uma a uma, de maneira que após a inserção e treinamento de uma nova camada, outra poderá ser inserida. A adição de uma nova camada deve preservar o aprendizado já obtido, para isso se utiliza um pré-processamento na camada recém inserida, com objetivo de adicionar progressivamente a influência da nova camada no treinamento global da rede. Esse pre-processamento é tido como primordial para que as novas camadas não atuem de forma negativa no aprendizado. Ou seja, fazendo com que o erro aumente. Após o final das inserções, a estrutura da CollabNet deverá ser similar à estrutura apresentada na Figura 6

Figura 6 – Estrutura básica da CollabNet



Fonte: Autor

A estrutura da CollabNet é dada conforme a Equação 3.1. Nesse caso, w representa os pesos de treinamento das camadas, $f^{(1)}$ é a função de ativação dos neurônios da primeira camada, $f^{(2)}$ da segunda, $f^{(2)}$ representa a função da terceira camada e fn de forma semelhante para as mais camadas. A última camada é denominada de camada de saída e a profundidade da rede é dada pelo comprimento total da cadeia de camadas (GOODFELLOW; BENGIO; COURVILLE, 2016).

$$f(x) = w \cdot f^{(3)}(w \cdot f^{(2)}(w \cdot f^{(1)}(x))) \tag{3.1}$$

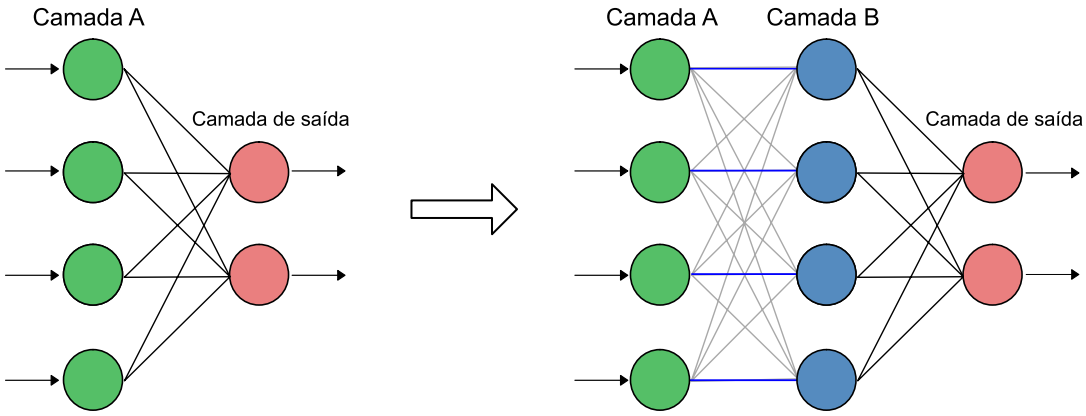
3.2 TREINAMENTO

A estratégia de treinamento proposta neste trabalho inicia-se de modo análogo ao método tradicional de uma rede MLP, com apenas uma camada escondida e a utilização do algoritmo

backpropagation (GLOROT; BENGIO, 2010). Passada essa primeira etapa do treinamento, quando o erro de saída da rede não decresce mais, então uma nova camada pode ser inserida na rede em treinamento. Essa inclusão, necessariamente, deve ser realizada uma a uma, haja vista que após a inclusão de uma nova camada é necessário realizar diversos procedimentos, comentados a seguir, visando à inclusão harmoniosa das novas camadas escondidas.

A abordagem proposta apresenta uma forma inovadora de inserção de novas camadas. Esse método tem como objetivo garantir que o erro de saída da rede não seja corrompido por novas camadas. Para isso, é necessário que a saída de cada neurônio da camada recém inserida seja exatamente o mesmo valor da saída do neurônio correspondente da camada anterior considerando todos os dados de entrada (Figura 7).

Figura 7 – Inserção de uma nova camada

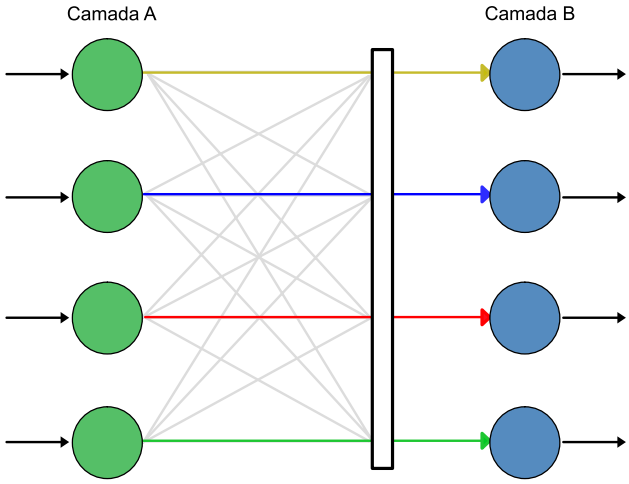


Fonte: Autor

Nesta proposta, a inicialização dos pesos entre as camadas A e B é realizada de forma aleatória. Entretanto, para que a saída da camada B seja desde o início exatamente a mesma saída existente na camada A, torna-se necessário providenciar um tratamento na saída da camada A, do contrário não há como garantir que a saída da camada B seja a mesma da camada A, pois a saída da camada B será alterada pelos pesos recém inicializados e pela função de ativação dos neurônios da camada B.

Para compensar a alteração devida aos pesos recém criados, este trabalho propõe realizar o tratamento dos dados entre as camadas A e B. Esse tratamento consiste de uma espécie de máscara, aqui chamada D, conforme apresentado na Figura 8. A máscara D modifica os valores que chegam nos neurônios da camada B, permitindo que a saída de cada neurônio seja exatamente igual a saída do neurônio correspondente na camada A.

Figura 8 – Máscara *D* entre uma camada antiga e a camada recém inserida



Fonte: Autor

A máscara *D* garante que cada neurônio da nova camada receba apenas a influência de seu neurônio correspondente da camada anterior. O funcionamento da máscara *D* é apresentado na Figura 8 e ocorre da seguinte forma: as conexões representadas por uma linha colorida indicam que o valor não é alterado pela máscara, ou seja, os neurônios da nova camada (B) recebem exatamente o valor de saída do neurônio correspondente na camada anterior (A). As conexões representadas por uma linha cinza têm seu valor anulado pela máscara, assim, o valor recebido dos neurônios adjacentes é nulo. Desta forma, o valor de entrada de cada neurônio da nova camada é exatamente o mesmo valor que sai do neurônio na camada A.

Para que a máscara *D* realize o filtro dito acima, é necessário haver uma multiplicação pelo inverso do valor do peso correspondente, de tal modo que o valor efetivamente processado pelo neurônio da camada B é exatamente o valor de saída do neurônio correspondente da camada A. Desta forma, é garantido que o treinamento da camada B iniciará exatamente do ponto onde a camada anterior parou. É garantido também que novas inserções não atrapalhem o aprendizado da rede como um todo.

Após a inclusão da camada B, parte-se, então, para o cálculo da entrada desta camada, conforme Equação 3.2, onde W_{hi} são os pesos entre as camadas A e B, D é a máscara e Y é a saída da camada A. O operador $.*$ significa uma multiplicação elemento a elemento e não uma multiplicação matricial normal.

$$net_h = (W_{hi} .* D) * Y \tag{3.2}$$

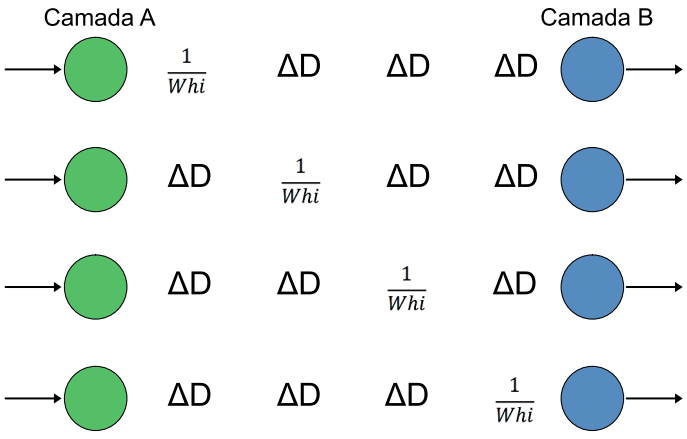
Além da máscara *D*, outra modificação proposta neste trabalho consiste em alterar a função de ativação dos neurônios da camada B para, em vez de sigmóide, ser a função identidade. Desta forma, o que entra nos neurônios da camada B torna-se saída desta camada. Assim, tem-se a garantia de que a saída de cada neurônio da camada B é exatamente a saída do neurônio correspondente na camada A.

Entretanto, as inovações propostas neste trabalho, da forma como estão, não permitem que a nova camada adquira um aprendizado, pois a saída da camada B será sempre igual a saída da camada A, não havendo, portanto, diminuição do erro de saída da rede por modificação dos novos pesos criados. Assim, após a inserção da camada B, quando o algoritmo de treinamento de rede é executado, tanto a máscara *D* quanto a função de ativação dos neurônios da camada B devem sofrer uma alteração, de modo a permitir que haja uma influência na saída da rede.

Contudo, uma retirada abrupta da máscara *D* e/ou a permuta da função de ativação tipo identidade para sigmóide promovem, por um lado, a possibilidade de aquisição de aprendizado pela camada B, mas por outro lado, providenciam uma elevação brusca no erro de saída da rede. Assim, faz-se necessário que haja uma transição suave e gradual da retirada da máscara *D* e da conversão da função identidade para sigmóide.

A transição mencionada no parágrafo anterior deve ocorrer durante a execução do treinamento. Assim, após um certo número de iterações (neste trabalho foi em torno de 300), os pesos entre as camadas A e B deverão promover a redução do erro de saída da rede, sem prejudicar o decaimento do erro promovido pelas camadas anteriores. A influência desses novos pesos no treinamento deve ser realizada transformando a máscara *D* em uma matriz com todos os elementos unitários. Essa transformação é realizada por meio de um ΔD que vária de 0 a 1, com velocidade definida via parametrização da inicialização da nova camada, conforme ilustrado na Figura 9. Desta maneira, pretende-se garantir que a perturbação provocada pela inserção de uma nova camada escondida na rede seja a menor possível.

Figura 9 – Alteração dos valores da máscara *D* por meio do ΔD



Fonte: Autor

Enquanto todos os elementos da máscara *D* váriam de modo uniforme, os elementos de sua diagonal principal váriam de acordo com os pesos aleatórios gerados na inicialização da nova camada a depender dos valores iniciais desses pesos, os valores da diagonal principal convergem mais rápido a um.

O funcionamento do algoritmo CollabNet é apresentado no pseudocódigo 2. No entanto,

a inclusão de uma nova camada nesta abordagem é dada de forma manual, pois o parâmetro que determina a estabilidade do erro no treinamento, descrito na linha 7, requer um estudo específico com objetivo de obter um desempenho quanto a base utilizada, o número de neurônios das camadas, a velocidade de alteração da variável c e os demais parâmetros. Nesta abordagem, a decisão de inclusão de uma nova camada fica a cargo da *expertise* do projetista.

Algoritmo 2: Pseudo-código da CollabNet

```
   Entrada: Um conjunto de n objetos de treinamento
   Saída: CollabNet com pesos ajustados
1  início
2      Inicializar os pesos
3      repita
4          para cada objeto x_i do conjunto de treinamento faça
5              Calcular valor da saída produzida pelo neurônio, f̂(x_i)
6              Calcular erro = y_i − f̂(x_i)
7              se erro > 0 and (erro não estabilizado) então
8                  Ajustar pesos do neurônio
9              senão se erro > 0 então
10                 x = saída camada_{i−1}
11                 Inclui nova camada
12             fim
13         fim
14     até erro = 0;
15 fim
```

Fonte: Autor

Para esta abordagem foi utilizada a função de ativação sigmóide, no entanto, sua utilização é realizada apenas quando a rede possui apenas uma camada escondida. Assim, a partir da inclusão da segunda camada escondida, foi necessário realizar uma pequena adaptação na função sigmóide, com a intenção de controlar, precisamente, a influência da nova camada no aprendizado da rede. Tal adaptação é apresentada na Eq. 3.3.

$$\phi_A(n) = \frac{1}{1+e^{-n}} * c + n * (1-c) \tag{3.3}$$

onde $\phi_A(n)$ é a saída do neurônio com a $sigmoide_A$, c é o fator de ponderação da função de ativação e n é a soma ponderada de todas as entradas sinápticas dos neurônios.

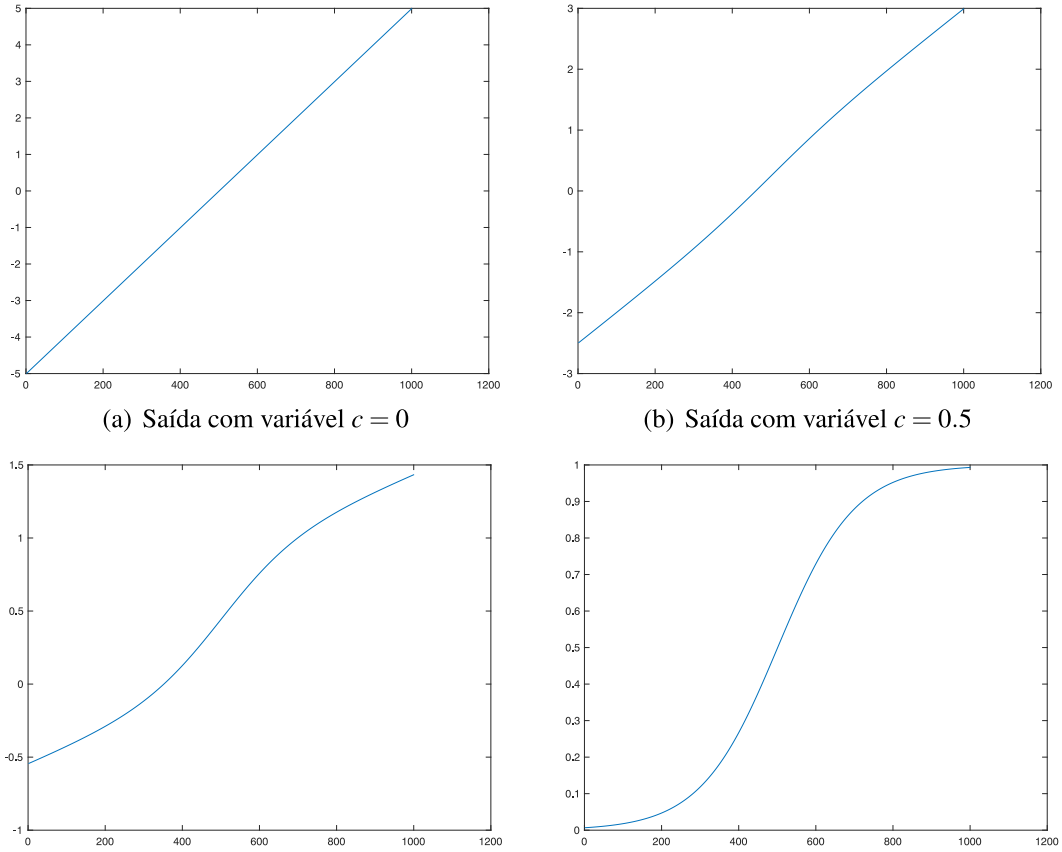
A adaptação realizada na Eq. 3.3 foi implementada pela necessidade de que a função de ativação seja a função identidade no inicio de treinamento de uma nova camada. Ao longo

do treinamento dessa camada, a função de ativação deve ser gradualmente convertida na função sigmóide, o que é promovido pela variável c, de modo que após uma certa quantidade de iterações, a função de ativação volta a ser apenas a função sigmóide tradicional. Assim, a variável c atua ponderando as funções identidade e sigmóide, transformando uma função de ativação de uma identidade no início do treinamento da nova camada, para uma função sigmóide ao final.

A inclusão da variável c na função de ativação (ϕ) sigmóide traz ao projetista o poder de controlar a influência de uma camada no aprendizado, no qual quanto mais próximo de 1 é o valor da variável c, maior é sua influência e mais próximo de 0, menor será tal influência. A utilização da variável c no treinamento é de grande importância para o método de inclusão de novas camadas, pois com esse artifício, é possível garantir que a influência da nova camada seja gradual, de acordo com que os pesos da nova camada vão adequando-se ao problema, uma vez que, por padrão, estes são gerados aleatoriamente para cada nova camada inserida.

A nova função de ativação denominada $sigmoide_A(\phi_A)$ é dada em função da variável c, onde nas primeiras iterações do treinamento, o valor de c é zero, garantindo que os novos pesos não sejam considerados imediatamente no treinamento (Figura 10 (a)).

Figura 10 – Função sigmóide alterada pelo método (ϕ_A)



(a) Saída com variável $c = 0$

(b) Saída com variável $c = 0.5$

(c) Saída com variável $c = 0.89$

(d) Saída com variável $c = 1$

Fonte: Autor

Ao passo que as iterações do treinamento avançam, o valor da variável c aproxima-se de 1 (Figura 10 (c)) e consequentemente a saída dessa camada influencia na redução do erro de saída da rede. Após a variável c alcançar o valor de 1, a camada B passa a funcionar com a função sigmóide somente (Figura 10 (d)) e seu processo de inclusão está, então, finalizado. Com a adição finalizada de mais uma camada escondida, continua-se o treinamento da rede completa do modo tradicional, tendo em vista que nesse momento do treinamento, os pesos da camada recém inserida estejam alinhados ao treinamento anterior.

Em decorrência da alteração na função de ativação sigmóide citada acima, foi necessário usar a derivação da função alterada no algoritmo *backprogation* (Eq. 3.4), para que o cálculo do gradiente seja correto.

$$\frac{\mathrm{d}\phi_A}{\mathrm{d}n} = c \cdot (y) \cdot (1 - y) + (1 - c) \tag{3.4}$$

onde y é a sigmoide tradicional e c é o fator de ponderação.

Com a utilização da máscara D e a função de ativação alterada com o uso da variável c, a inserção de uma nova camada não interferiu negativamente no aprendizado, como poderá ser visto no capítulo seguinte com a apresentação dos resultados obtidos nos experimentos.

4 EXPERIMENTOS

A demonstração da aplicação da CollabNet foi dada em uma tarefa de reconhecimento de padrões. A base utilizada foi a *Wisconsin Breast Cancer Dataset* (WOLBERG; STREET; MANGASARIAN, 2011), retirada do repositório de aprendizado de máquinas da Universidade da Califórnia em Irvine (UCI). A referida base possui informações de 669 registros de tumores de mama, possuindo duas classes identificadas como tumores malignos (M) e benignos (B), cada uma com dez características reais calculadas para cada núcleo celular: raio, textura, perímetro, área, suavidade (variação local no comprimento do raio), compacidade ($\frac{perimetro^2}{area-1.0}$), concavidade (gravidade das porções côncavas do contorno), pontos côncavos, simetria e dimensão fractal.

Para a referida base de dados, foram testadas diversas configurações da rede proposta variando diversos parâmetros de inicialização, tanto para rede quanto para as novas camadas: quantidade de neurônios e épocas, taxa de aprendizado, valor e momento do incremento da variável c e da máscara D (ΔD) e o comportamento dos pesos no treinamento. Entretanto, a quantidade de neurônios nas camadas escondidas é um parâmetro básico que diz respeito a estrutura da rede. Esse parâmetro é definido no momento de criação da rede, ficando imutável a partir da inserção da segunda camada escondida, garantindo que cada nova camada escondida altere apenas a profundidade da rede e não a sua largura.

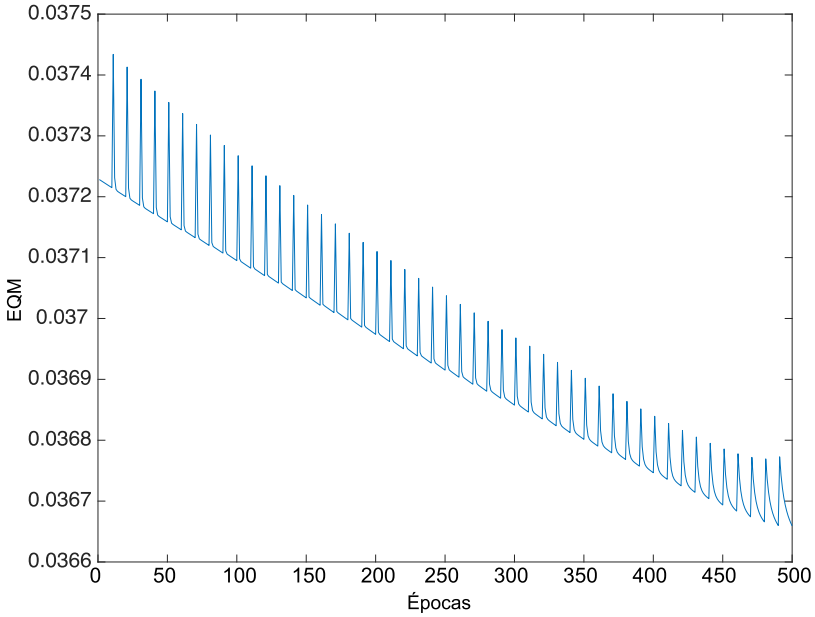
4.1 PARÂMETRIZAÇÃO

Os parâmetros de treinamento foram estimados empiricamente e sempre com a inclusão de uma nova camada. A taxa de aprendizagem, a quantidade de épocas, o comportamento dos pesos da nova camada em relação a sua inicialização são parâmetros essenciais da inserção de uma nova camada. E por fim, têm-se os parâmetros referentes ao comportamento da variável c e da máscara D, definindo informações quanto às velocidades de variação destas variáveis no treinamento.

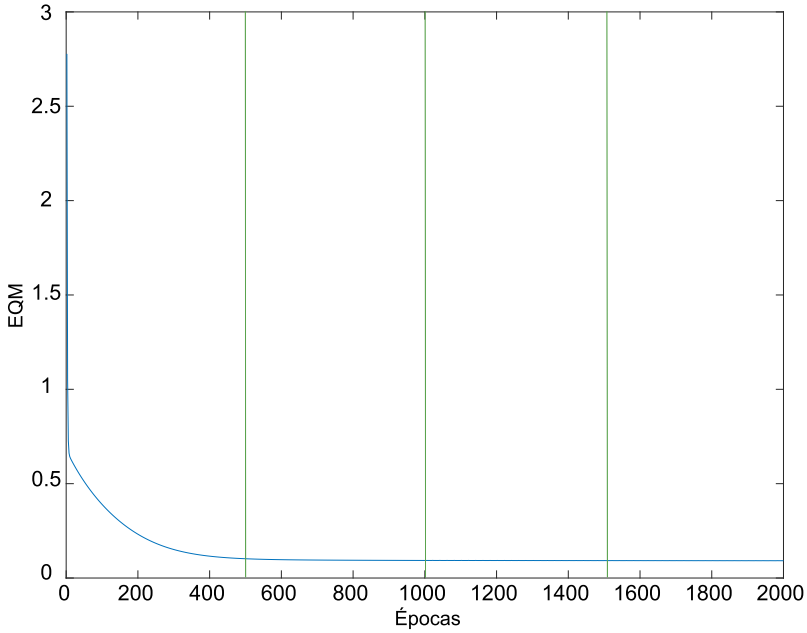
Diante dos diversos parâmetros da rede, a variável c merece uma explanação mais abrangente, que neste projeto desempenha uma função especial. Essa variável está diretamente relacionada a inclusão de uma nova camada, bem como sua transição, fazendo com que uma camada recém inserida, sem utilidade para o aprendizado da rede, possa tornar-se um elemento de importância a esse aprendizado, conforme é apresentado no Capítulo 3. A variável c tem a responsabilidade de controlar a influência de uma nova camada no treinamento, sendo que tal influência é uma grandeza diretamente proporcional ao valor de c, ou seja, quanto mais próximo c está de seu valor máximo (1), maior é a influência da nova camada no treinamento. Portanto, o controle de c é o grande desafio desta proposta e a maneira que o valor desta variável aumenta no decorrer do treinamento da nova camada necessita ser parametrizada de forma individual. A parametrização ocorreu de forma empírica, sendo escolhidos valores entre 0,001 e 0,003.

O ajuste do incremento de c deve ser cuidadosamente observado pelo projetista, pois sua parametrização correta tem influência direta no comportamento do erro quadrático médio (EQM). A Figura 11 (a) apresenta uma visão ampliada da última camada do treinamento apresentado na Figura 11 (b), onde nessa inclusão foi definido um incremento da variável c relativamente alto, aproximadamente 0,3 a cada iteração. Nela é possível perceber as perturbações no EQM em cada iteração da variável c. Esse fenômeno é explicado por uma variação alta da variável c e pelos pesos que ainda não estão próximos da convergência do valor final.

Figura 11 – Comportamento do EQM de acordo com as variações de c



(a) Variável c com valor de incremento alto



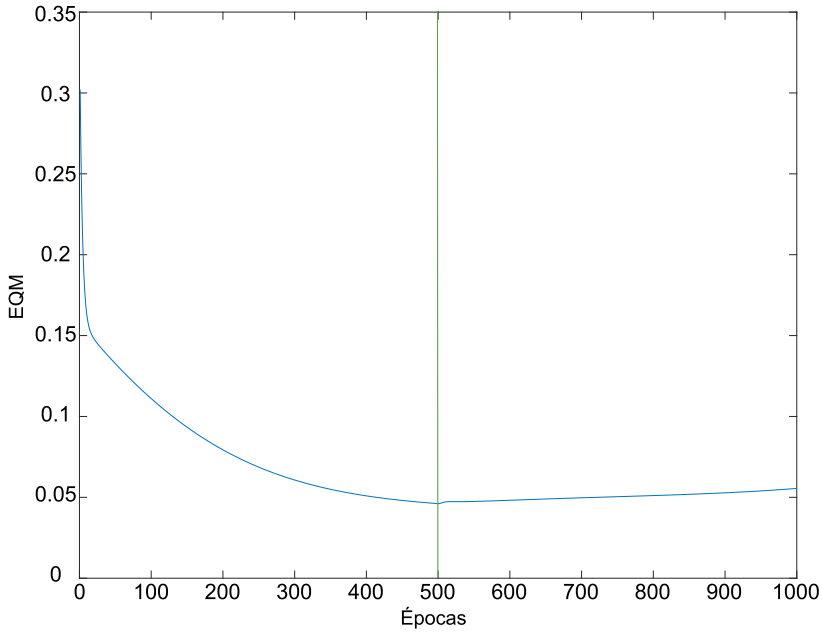
(b) Variável c com valor de incremento baixo

Fonte: Autor

Na Figura 11 (b), é apresentado o EQM após a inclusão de uma nova camada com incremento da variável c relativamente baixo. Neste caso, a influência da nova camada é lenta e seria necessário aumentar a quantidade de épocas de treinamento de cada camada.

Na Figura 12, é apresentado outro comportamento indesejado do EQM no treinamento. Nesse caso, o parâmetro c foi configurado para ser incrementado a cada época de treinamento. Essa configuração tem a tendência de fazer com que o erro suba gradualmente.

Figura 12 – Variável c configurada para receber um incremento a cada época de treinamento

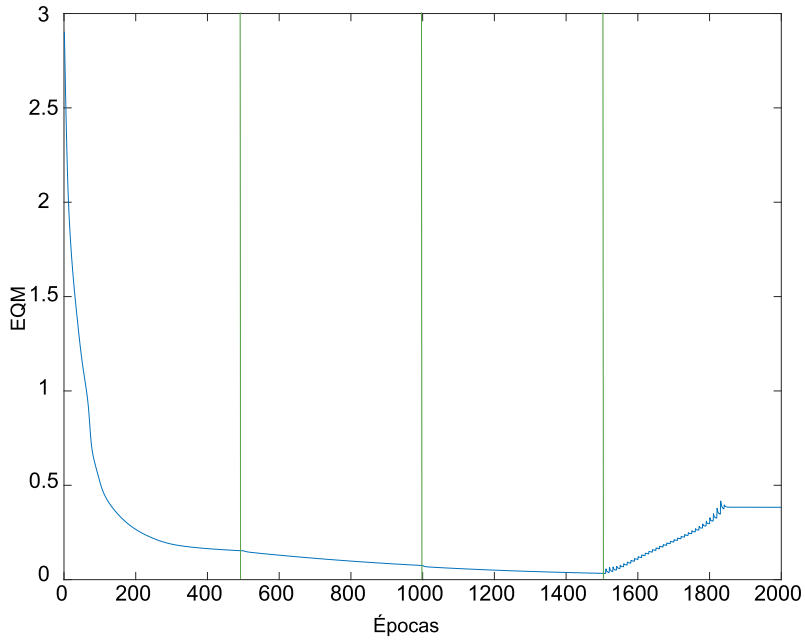


Fonte: Autor

Igualmente a variável c, a variação da máscara D também desempenha um papel importante na inclusão e controle de novas camadas, necessitando que seja observada no ato de inserção de novas camadas, tendo em vista que ΔD possui sua variação intrínseca ao valor do incremento de c e está diretamente relacionado à transformação da máscara D. A parametrização correta de ΔD é importante ao treinamento, pois esse parâmetro em conjunto com a variável c controla a influência da nova camada no treinamento global. A máscara D deve ser iniciada com 0, a excessão da diagonal principal e sua atualização é realizada simultâneamente com a varável c pelo valor ΔD.

Uma configuração não harmoniosa desses parâmetros gera um comportamento indesejado do EQM, e consequentemente, traz prejuízo ao treinamento. Como exemplo desse comportamento indesejável, são apresentados casos em que ΔD é alto (Figura 13) e baixo (Figura 14), respectivamente. Esse parâmetro diz respeito à velocidade que a máscara D fica invisível ao treinamento, ou seja, o processo de inclusão da nova camada foi completado.

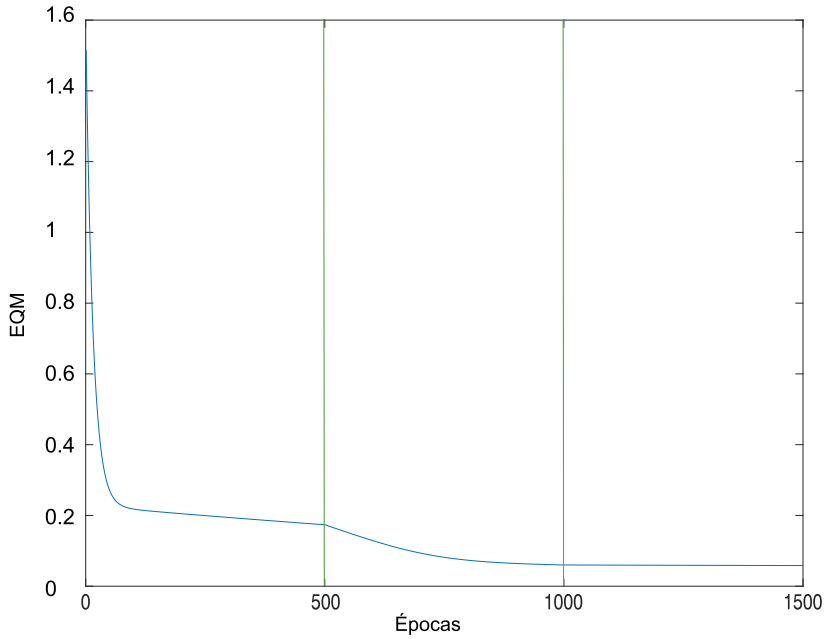
Figura 13 – ΔD com valor alto



Fonte: Autor

A Figura 13 apresenta um comportamento indesejado na última camada, causado pelo velocidade de convergência da máscara D. Na Figura 14, é apresentado outro comportamento indesejado, onde a influência da nova camada é praticamente nula, pois a taxa de atualização de ΔD foi relativamente pequena.

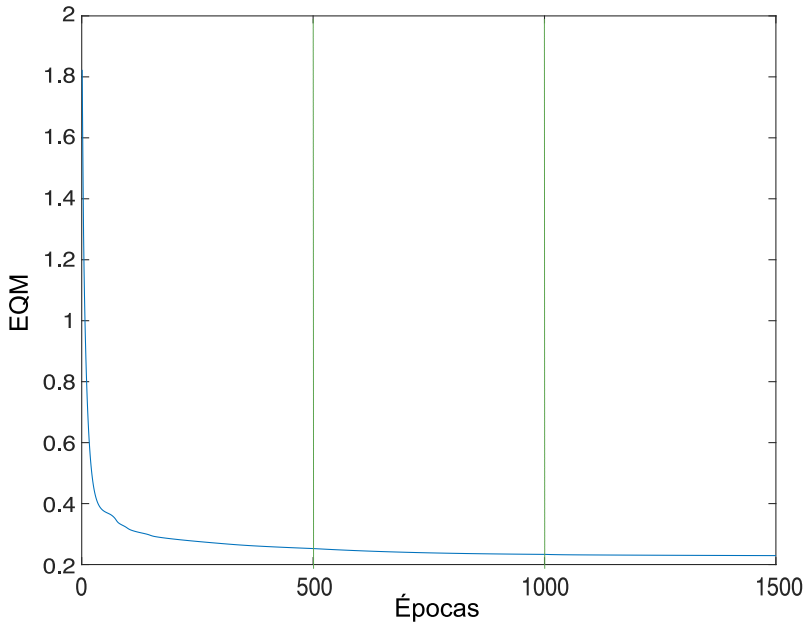
Figura 14 – ΔD com valor baixo



Fonte: Autor

Outro grande desafio na parametrização da CollabNet, assim como em qualquer abordagem baseada em MLP, é a definição da taxa de aprendizado. Esse parâmetro é diretamente relacionado à velocidade de aprendizado da rede. Na CollabNet, a taxa de aprendizado pode ser distinta para cada camada. Desta forma, é necessário que o projetista tenha a sensibilidade e a destreza para definir o valor da taxa de aprendizado para cada camada. A Figura 15 ilustra a saída da CollabNet, com três camadas escondidas e o valor da taxa de aprendizado relativamente baixo ($5x10^{-}5$).

Figura 15 – EQM com valor da taxa de aprendizado baixo

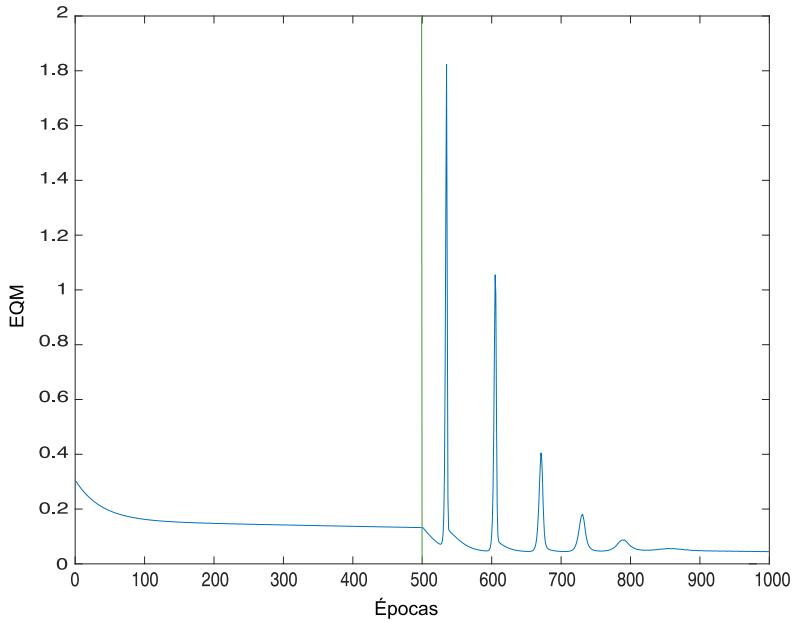


Fonte: Autor

Nesse exemplo, é apresentado o comportamento do EQM de saída com uma taxa de aprendizado baixa. Assim, o EQM tende a dar continuidade a saída da rede em camadas anteriores, sem promover nenhuma melhora no decaimento do erro de saída.

Na Figura 16, é apresentado o comportamento do erro com a inclusão de uma nova camada, agora, com taxa de aprendizado tendo um valor relativamente alto (0.1). É possível observar que a tendência do erro, nesse caso, é de inicialmente baixar e logo após aumentar bruscamente a cada incremento de c. Esse comportamento é explicado pelo fato de que, com a taxa de aprendizado alta em um momento onde os pesos da nova camada ainda não estão em conformidade com os pesos já treinados em camadas anteriores, gerando assim uma perturbação maior no EQM. Tal fato pode ser ainda percebido, observando a parte final gráfico, no qual os pesos já estão mais adequados ao restante do treinamento, assim a perturbação do EQM a cada novo incremento de c é menor.

Figura 16 – EQM com valor da taxa de aprendizado alta

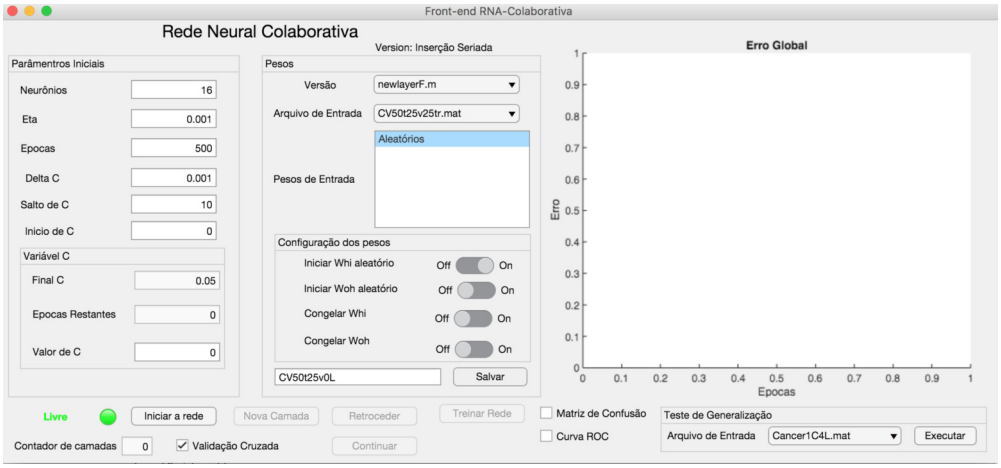


Fonte: Autor

4.2 METODOLOGIA

A rede proposta neste trabalho foi implementada em ambiente *Matlab 2016a* sem nenhum pacote adicional. A implementação da rede foi realizada sem o uso das funções da biblioteca de redes neurais do Matlab, no intuito de obter um maior controle dos procedimentos executados por ela. Diante das diversas possibilidades de configuração da rede proposta, houve a necessidade de criação de uma interface de usuário (GUI), disposta na Figura 17. A GUI foi importante para execução dos treinamentos e testes de validação, pois com sua utilização foi possível agilizar a parametrização e execução da rede.

Figura 17 – Interface do usuário

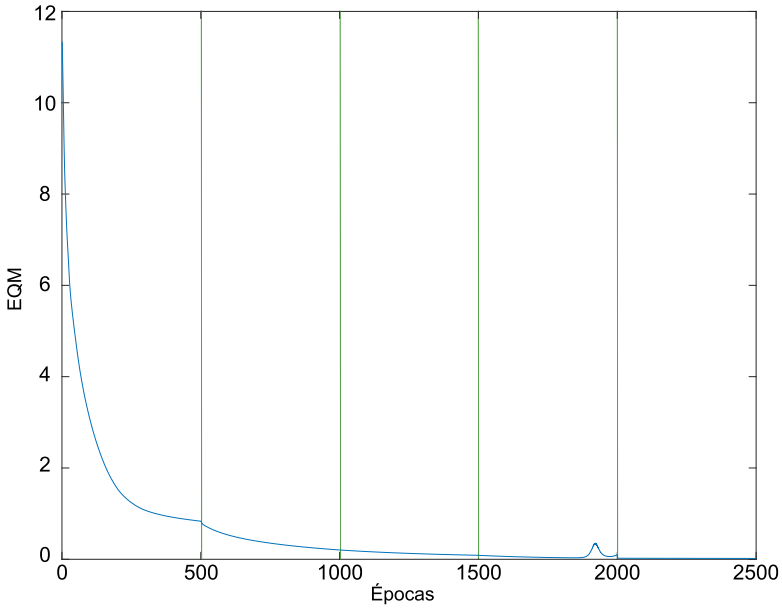


Fonte: Autor

Para treinamento e testes da rede, a base de dados foi dividida em duas partes. Uma parte contendo 75% da base dados foi usada para o treinamento e o restante, para testes. Ao final dos testes, foi promovida uma validação cruzada dos dados para verificação da independencia do resultado com relação aos dados. Após vários testes com diversas configurações, obteve-se um melhor desempenho na base selecionada, com a utilização de 16 neurônios em todas as camadas escondidas, uma taxa de aprendizado variando entre $\eta = 10^{-3}$ a $\eta = 10^{-5}$. A variável c iniciada em 0 com um salto variando entre 10^{-2} e 10^{-3} a cada 20 épocas, o ΔD variando na mesma taxa que c, a função de ativação $sigmoide_A$ nas camadas intermediárias e a função linear na camada de saída.

Com essa configuração, a CollabNet obteve um decaimento do erro de saída de forma mais eficiente, com decaimento do erro permanente, quando comparado ao treinamento da MLP sem o método proposto por este trabalho, conforme é apresentado na Figura 18. A taxa de acerto nessa configuração foi de $95,7\%$.

Figura 18 – Decaimento do EQM no cenário citado acima



Fonte: Autor

4.3 MÉTRICAS DE VALIDAÇÃO

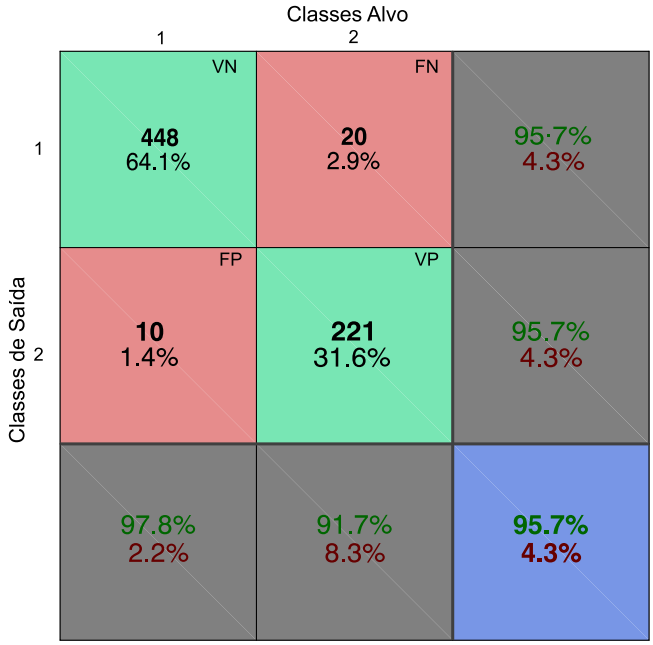
Neste trabalho, foram utilizadas como métricas de avaliação a matriz de confusão e a *receiver operating characteristic* - ROC. Em posse da matriz de confusão, é possível verificar o desempenho da rede na tarefa de classificação de padrões, independentemente da classe. A Figura 19 apresenta a matriz de confusão obtida a partir do experimento citado nessa seção, finalizado com cinco camadas escondidas. Nessa figura, a classe 1 representa os casos benignos e a classe 2 representa os casos malignos, as duas primeiras células diagonais mostram o número e a porcentagem de classificações corretas pela CollabNet, ou seja, 448 biópsias foram corretamente

classificadas como benignos (Verdadeiro Negativo - VN) e 221 casos foram corretamente classificados como malignos (Verdadeiro Positivo - VP), correspondendo respectivamente a 64,1% e 31,6% das 699 biópsias. 20 amostras (2.9% do total) das biópsias malignas foram incorretamente classificadas como benignas (Falso Positivo - FP). Da mesma forma, 10 biópsias (1,7% dos dados) foram incorretamente classificados como malignas (Falso Negativo - FN).

Das 468 previsões benignas, 95,7% foram corretas e 4,3%, erradas. Das 231 predições malignas, coincidentemente, 95,7% foram corretas e 4,3%, erradas. Dos 458 casos benignos, 97,8% dos casos foram corretamente preditos como benignos e 2,2% foram preditos como malignos. Dos 241 casos malignos, 91,7% foram classificados corretamente como malignos e 8,3% como benignos.

No geral, 95,7% das previsões foram corretas e 4,3% foram classificações erradas.

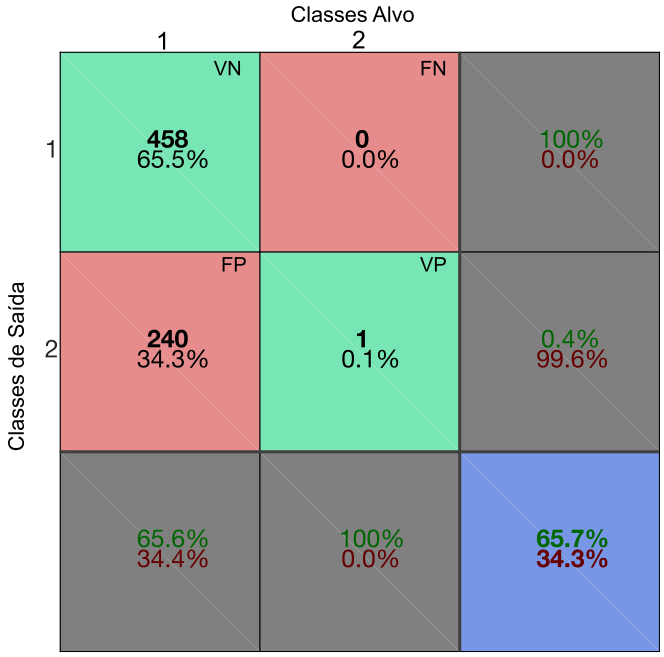
Figura 19 – Matriz de confusão ao final do treinamento



Fonte: Autor

A CollabNet trabalha com sucessivas inserções de camadas escondidas. A cada nova camada inserida, espera-se que os resultados da rede sejam sempre melhorados. Diante disso, a Figura 20 apresenta a matriz de confusão da CollabNet ao final do treinamento com apenas uma camada escondida. Nesse momento, a taxa de acertos foi de 65,7%.

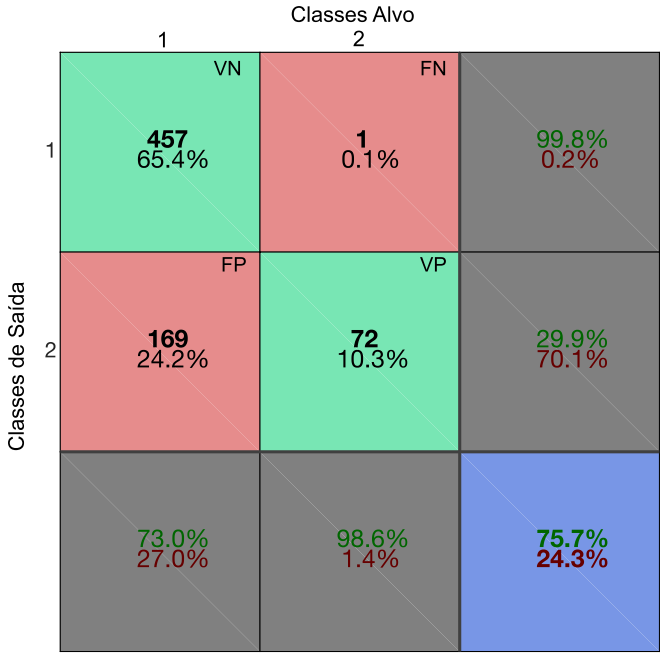
Figura 20 – Matriz de confusão com 1 camada



Fonte: Autor

Na Figura 21, é apresentada a matriz de confusão do treinamento com apenas 2 camadas escondidas. Nela, houve um aumento de previsões corretas, chegando a 75,7%.

Figura 21 – Matriz de confusão com 2 camadas

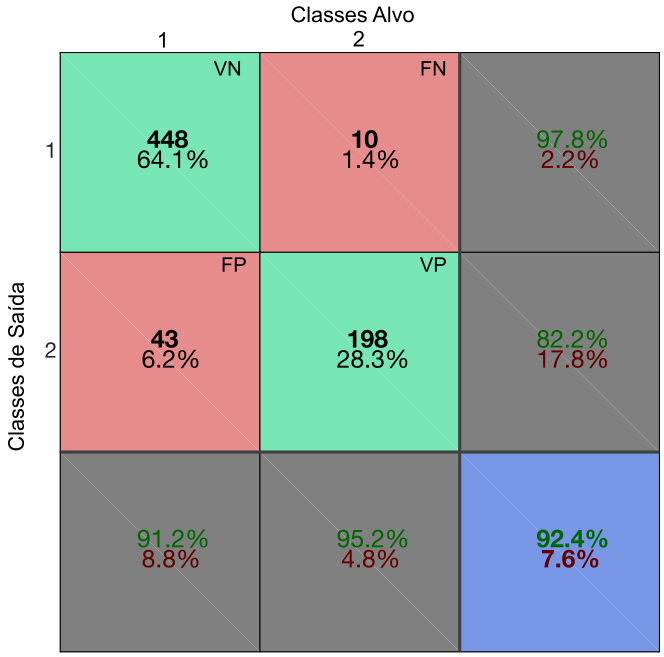


Fonte: Autor

A Figura 22 apresenta a matriz de confusão com três camadas escondidas e uma taxa de

acertos de 92,4%, mostrando que a inserção de uma nova camada melhora o acerto.

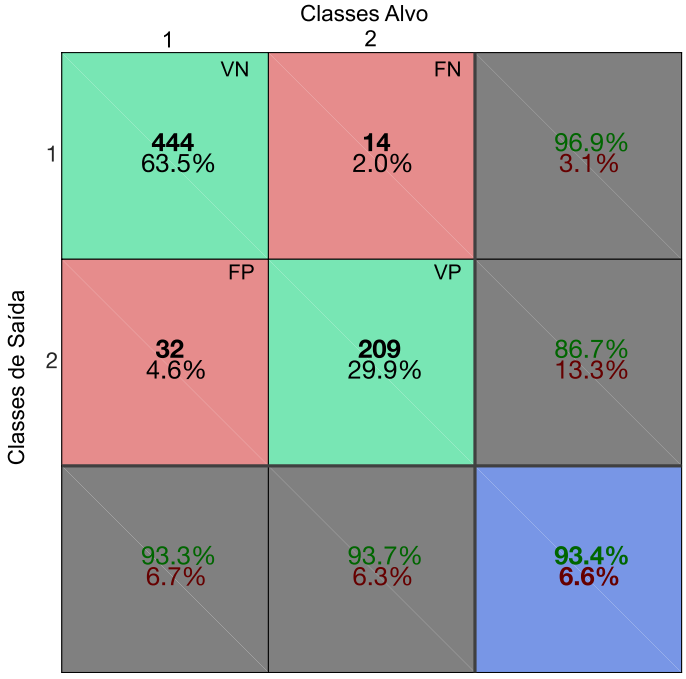
Figura 22 – Matriz de confusão com 3 camadas



Fonte: Autor

Por fim, a Figura 23 apresenta a matriz confusão da CollabNet com quatro camadas escondidas, uma menos que o treinamento final, apresentado na Figura 19. Percebe-se novamente uma melhora na taxa de acerto.

Figura 23 – Matriz de confusão com 4 camadas

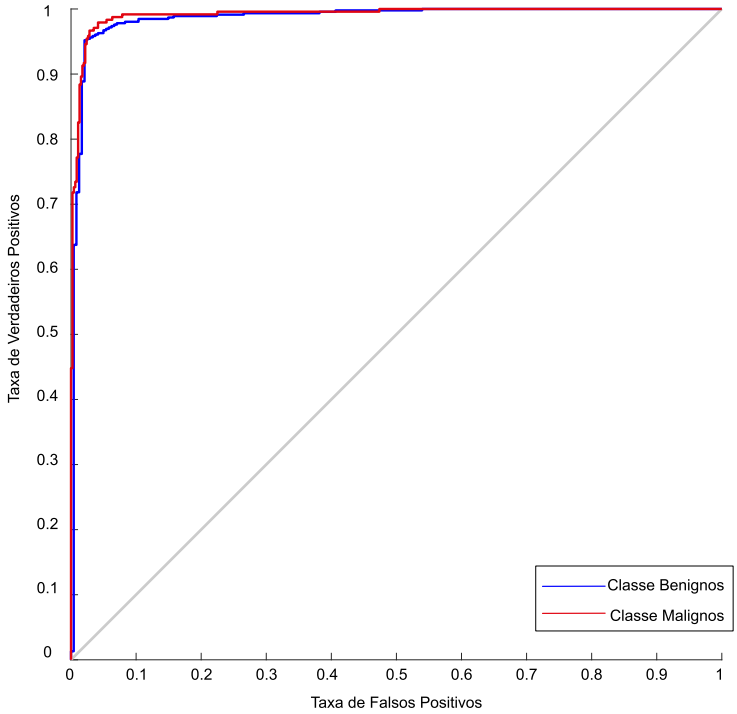


Fonte: Autor

Outra métrica utilizada na avaliação da CollabNet é a ROC, ou curva ROC, onde para cada classe desse classificador, tem-se valores no intervalo de [0,1] para cada saída. Para cada classe, são calculados dois valores, a Taxa de VP ou sensibilidade e a Taxa FP ou especificidade. Portanto, a sensibilidade é a capacidade do sistema de predizer corretamente a condição para casos que realmente a possuem, enquanto que a especificidade é a capacidade do sistema de predizer corretamente os casos que não possuem determinada condição.

A Figura 24 ilustra a curva ROC para a configuração da CollabNet apresentada nesta seção, onde, quanto mais próximas do canto superior esquerdo estão as linhas do gráfico, melhor é a classificação.

Figura 24 – Curva ROC



Fonte: Autor

Nesse gráfico é possível perceber que ambas as classes, tumores malignos (azul) e benignos (vermelho), possuem suas curvas próximas do canto superior esquerdo, demonstrando assim, a eficiência do método na classificação da base de dados selecionada. No entanto, uma aproximação maior das curvas no ponto de interesse da curva ROC deve ser objeto de busca constante em qualquer algoritmo de aprendizado de máquinas. Desta forma, ajustes nos parâmetros e mais testes de validação são importantes na busca de melhores resultados do classificador.

5 CONCLUSÃO

Técnicas de aprendizado profundo estão trazendo grandes avanços na resolução de problemas que resistiram às melhores tentativas da comunidade de inteligência artificial por muitos anos. A aplicação de *deep learning* em estruturas de dados com alta dimensionalidade, que estão presentes em diversos domínios da ciência, das empresas e do governo, tem demonstrado um futuro promissor. *Deep learning*, além de vencer diversas competições de reconhecimento de imagem e voz, superou também técnicas conceituadas de aprendizado de máquinas, nas mais diversas áreas de aplicação, como reconhecimento de linguagem natural, análise de sentimentos, análise de dados de aceleradores de partículas, tradução de idiomas, etc.

Deep learning é uma área com futuro promissor e tende a crescer e ter resultados cada vez melhores em muitas aplicações. O fato de os algoritmos de *deep learning* tirarem proveito do aumento do poder computacional e dos dados disponíveis é um fator determinante para este crescimento. Outro fator determinante para a consolidação do *deep learning* com uma subarea de conhecimento promissora é o crescente número de novos algoritmos e arquiteturas de aprendizagem que estão sendo desenvolvidos. Desta maneira, este trabalho propõe uma arquitetura de treinamento alterando o método tradicional de empilhamento de *autoencoders*, no entanto, esta abordagem não degrada o erro de saída da rede por promover uma alteração na entrada de dados em cada nova camada inserida. Tendo assim, um controle maior do treinamento, fazendo com que o treinamento de uma nova camada sempre se inicie do ponto onde a rede parou o treinamento com uma camada a menos.

O presente trabalho teve como proposta apresentar uma inovação no que se refere à inserção e o treinamento de novas camadas totalmente conectadas, com a intenção de realizar um pré-processamento nos dados de entrada das camadas recém inseridas, com a finalidade de manter sob controle a influência das novas camadas no treinamento. Para isso, foi realizada uma alteração na função sigmóide possibilitando o controle da influência da nova camada no treinamento global da rede.

5.1 CONTRIBUIÇÕES DESTE TRABALHO

A primeira contribuição deste trabalho foi apresentar uma novo método de treinamento profundo em redes neurais do tipo *feedforward*. Esta nova estratégia possui a característica de inserir novas camadas escondidas em uma estrutura de modo colaborativo, ou seja, esta nova camada irá continuar o aprendizado já adquirido pela rede e sua inserção deverá perturbar minimamente a decida do erro.

Outro ponto importante é a capacidade de inserção de camadas escondidas a medida que se achar necessário. Isto é um contraponto com o que ocorre em grande parte das redes profundas, nas quais a profundidade da rede é um parâmetro estático e definido no início do

treinamento.

Outra contribuição foi a formulação da função de ativação $sigmoide_A(\phi_A)$. Esta função de ativação é derivada da sigmóide tradicional e tem a capacidade adicional de controlar a influência de uma determinada camada escondida no treinamento de uma rede neural profunda.

5.2 TRABALHOS FUTUROS

A partir desta pesquisa é possível visualizar diversas abordagens com objetivo de explorar lacunas e/ou aprimorar algum aspecto abordado na CollabNet. Dentre os inúmeros desdobramentos possíveis, é possível destacar esforços que visem à concepção de novas formas de tratamento da variável c, por exemplo, fazer com que c varie em função de outros fatores e não somente em função das iterações da rede. A variação de c ainda pode ter outros desdobramentos, como por exemplo, realizar o incremento por meio de uma função exponencial ou qualquer outra função linear ou não-linear.

Outro desdobramento interessante desta pesquisa seria a utilização de algum algoritmo inteligente para identificar o momento ideal para inclusão de uma nova camada.

Enfim, este projeto é uma semente promissora de uma estrutura de RNA profunda e, com alguns aprimoramentos, poderá vir a ser um importante método de aprendizado de máquinas.

Referências

ALMOUSLI, H. *Recognition of Facial Expressions with Autoencoders and Convolutional-Nets*. Tese (Doutorado) — Université de Montréal, 2014. Disponível em: <http://hdl.handle.net/1866/10688>. 22

BABRI, H.; TONG, Y. Deep feedforward networks: application to pattern recognition. In: *Proceedings of International Conference on Neural Networks (ICNN'96)*. IEEE, 1996. v. 3, p. 1422–1426. Disponível em: <http://ieeexplore.ieee.org/document/549108/>. 18

BADRINARAYANAN, V.; KENDALL, A.; CIPOLLA, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. Disponível em: <http://mi.eng.cam.ac.uk/projects/segnet/.> 13

BENGIO, Y. Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, v. 2, n. 1, p. 1–127, 2009. ISSN 1935-8237. 17, 24

BENGIO, Y. et al. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems 19 (NIPS'06)*, 2006. NULL. Disponível em: <https://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf>. 14

de Almeida Neto, A. *Aplicações de Múltiplas Redes Neurais em Sistemas Mecatrônicos*. 155 p. Tese (Doutorado) — Instituto Tecnologico da Aeronáutica, 2013. 19

DENG, L.; YU, D. *Deep Learning Methods and Applications*. [S.l.]: Foundations and Trends in Signal Processing, 2013. ISBN 9781601988140. 17, 23, 24, 25

GAMA, J. et al. *Inteligência artificial: uma abordagem de aprendizado de máquina*. [S.l.]: Grupo Gen - LTC, 2015. ISBN 9788521618805. 20

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, v. 9, p. 249–256, 2010. ISSN 15324435. 29

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. 1ª. ed. Montreal: MIT Press Book, 2016. 800 p. Disponível em: <http://www.deeplearningbook.org>. 16, 18, 19, 23, 24, 28

GOODFELLOW, I. et al. Generative adversarial nets. In: *Advances in Neural Information Processing Systems 27*. [s.n.], 2014. p. 2672–2680. ISSN 10495258. Disponível em: <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>. 12, 19

HAYKIN, S. *Neural networks: a comprehensive foundation*. 2 ed. ed. New York: Prentice Hall, 1999. 938 p. 16, 19, 20, 22

HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. *Science*, v. 313, n. 5786, p. 504–507, 2006. ISSN 0036-8075, 1095-9203. 14

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing*, v. 70, n. 1, p. 489 – 501, 2006. ISSN 0925-2312. Neural Networks. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925231206000385>. 14

KRIZHEVSKY, A.; HINTON, G. E. Using very deep autoencoders for content-based image retrieval. In: *ESANN 2011, 19th European Symposium on Artificial Neural Networks*. Bruges, Belgium: [s.n.], 2011. p. 27–29. 24

LAROCHELLE, H. et al. Exploring Strategies for Training Deep Neural Networks. *Journal of Machine Learning Research*, v. 1, p. 1–40, 2009. Disponível em: <http://www.jmlr.org/papers/volume10/larochelle09a/larochelle09a.pdf>. 17

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, 2015. ISSN 0028-0836. 16, 17, 22

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967. p. 281–297. Disponível em: <https://projecteuclid.org/euclid.bsmsp/1200512992>. 14

MAKHZANI, A.; FREY, B. k-sparse autoencoders. *Internacional Conference on Learning Representations (ICLR)*, 12 2013. Disponível em: <http://arxiv.org/abs/1312.5663>. 14

NETANYAHU, N. S. Deeppainter : Painter classification using deep convolutional autoencoders. In: *International Conference on Arti cial Neural Networks (ICANN)*. [S.l.: s.n.], 2016. v. 9887, n. September, p. 1–8. 13

PERONA, P.; FINK, M. *The Full Images for Natural Knowledge Caltech Office DB*. Californa, 2013. 22

ROMERO, A. et al. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6, 12 2014. Disponível em: <http://arxiv.org/abs/1412.6550>. 14

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, oct 1986. ISSN 0028-0836. Disponível em: <http://www.nature.com/doifinder/10.1038/323533a0>. 20

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, v. 61, p. 85–117, oct 2014. ISSN 08936080. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0893608014002135>. 16

SUN, K. et al. Generalized extreme learning machine autoencoder and a new deep neural network. *Neurocomputing*, Elsevier, v. 230, n. December 2016, p. 374–381, 2017. ISSN 18728286. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2016.12.027>. 13, 14

TUR, G. et al. Towards deeper understanding: Deep convex networks for semantic utterance classification. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, p. 5045–5048, 2012. ISSN 15206149. 25

VINCENT, P.; LAROCHELLE, H. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion pierre-antoine manzagol. *Journal of Machine Learning Research*, v. 11, p. 3371–3408, 2010. 13

WANG, H.; HUANG, H.; MAKEDON, F. Emotion detection via discriminant laplacian embedding. *Univers. Access Inf. Soc.*, Springer-Verlag, Berlin, Heidelberg, v. 13, n. 1, p. 23–31, mar. 2014. ISSN 1615-5289. Disponível em: <http://dx.doi.org/10.1007/s10209-013-0312-5>. 14

WOLBERG, W. H.; STREET, W. N.; MANGASARIAN, O. L. *UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set*. http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29, 2011. Disponível em: <http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)>. 15, 35

XU, R.; WUNSCH II, D. Survey of clustering algorithms. *Trans. Neur. Netw.*, IEEE Press, Piscataway, NJ, USA, v. 16, n. 3, p. 645–678, maio 2005. ISSN 1045-9227. Disponível em: <https://doi.org/10.1109/TNN.2005.845141>. 14

YU, D.; DENG, L. Deep convex net: A scalable architecture for speech pattern classification. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, n. August, p. 2285–2288, 2011. ISSN 19909772. 13, 24