Instituto Federal Catarinense
Curso de Bacharelado em Ciência da Computação
Campus Blumenau

GABRIEL EDUARDO LIMA

APRENDIZADO PROFUNDO PARA CLASSIFICAÇÃO DE IMAGENS: EXPLORANDO O USO CONJUNTO DE REDES NEURAIS PARA CLASSIFICAÇÃO E REMOÇÃO DE RUÍDO EM IMAGENS DE DÍGITOS MANUSCRITOS

Blumenau
2023

GABRIEL EDUARDO LIMA

APRENDIZADO PROFUNDO PARA CLASSIFICAÇÃO DE IMAGENS: EXPLORANDO O USO CONJUNTO DE REDES NEURAIS PARA CLASSIFICAÇÃO E REMOÇÃO DE RUÍDO EM IMAGENS DE DÍGITOS MANUSCRITOS

Trabalho de Conclusão de Curso submetido ao Curso de Bacharelado em Ciência da Computação do Instituto Federal Catarinense — *Campus* Blumenau para a obtenção do título de Bacharel em Ciência da Computação.
Orientador: Ricardo de la Rocha Ladeira, Me.
Coorientador: Eder Augusto Penharbel, Me.

Blumenau
2023

GABRIEL EDUARDO LIMA

APRENDIZADO PROFUNDO PARA CLASSIFICAÇÃO DE IMAGENS EXPLORANDO O USO CONJUNTO DE REDES NEURAIS PARA CLASSIFICAÇÃO E REMOÇÃO DE RUÍDO EM IMAGENS DE DÍGITOS MANUSCRITOS

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo curso de Bacharelado em Ciência da Computação do Instituto Federal Catarinense – *Campus* Blumenau.

autenticação eletrônica na folha de assinaturas

Prof. Ricardo de la Rocha Ladeira, Me.

Orientador – IFC *Campus* Blumenau

BANCA EXAMINADORA

autenticação eletrônica na folha de assinaturas

Prof. Hylson Vescovi Netto, Dr.

IFC *Campus* Blumenau

autenticação eletrônica na folha de assinaturas

Paulo César Rodacki Gomes, Dr.

IFC *Campus* Blumenau

Blumenau

2023

LISTA_ASSINATURAS Nº 321/2023 - CCCOMP/BLU (11.01.09.22)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 14/12/2023 09:36)
HYLSON VESCOVI NETTO
PROFESSOR ENS BASICO TECN TECNOLOGICO
CCCOMP/BLU (11.01.09.22)
Matrícula: ###100#1

(Assinado digitalmente em 14/12/2023 13:55)
PAULO CESAR RODACKI GOMES
PROFESSOR ENS BASICO TECN TECNOLOGICO
CGE/BLU (11.01.09.01.03.07)
Matrícula: ###299#3

(Assinado digitalmente em 15/12/2023 00:04)
RICARDO DE LA ROCHA LADEIRA
PROFESSOR ENS BASICO TECN TECNOLOGICO
CGE/BLU (11.01.09.01.03.07)
Matrícula: ###779#0

*Visualize o documento original em https://sig.ifc.edu.br/documentos/ informando seu número: **321**, ano: **2023**, tipo: **LISTA_ASSINATURAS**, data de emissão: **14/12/2023** e o código de verificação: **ba449fcd6f***

Dedico este trabalho a Deus. Sem ele nada seria possível.

AGRADECIMENTOS

A Deus, pela minha vida e pela força para persistir na busca de meus objetivos.
Aos meus pais, pelo amor e apoio ao longo de toda a trajetória.
Aos professores, pela dedicação ao ensino com a qual nortearam meu aprendizado.
Aos meus colegas de curso, pelo convívio e troca de experiências.
Ao Instituto Federal Catarinense, em especial ao *Campus Blumenau*, pelas oportunidades.

[...] inteligência artificial, a ciência de fazer máquinas realizarem coisas que exigiriam inteligência se fossem feitas por homens (MINSKY, 1968, tradução nossa).

RESUMO

A Inteligência Artificial é o campo da Ciência da Computação que busca compreender e desenvolver máquinas inteligentes, capazes de tomar ações eficientes para solucionar problemas em cenários desconhecidos. Dentre seus subcampos está o Aprendizado Profundo, representado por algoritmos inspirados no cérebro biológico para o desenvolvimento de redes neurais artificiais. A sua relevância está na possibilidade de aplicação a diferentes problemas em áreas diversas. Como exemplo, e foco do trabalho, cita-se a classificação de imagens — caracterizada pela necessidade de atribuir um (ou mais) rótulo(s) a uma imagem com base na informação que ela representa. Todavia, é observado que redes neurais são negativamente afetadas pela presença de ruído nos dados. A redução desse impacto pode ser obtida pela aplicação de métodos de pré-processamento visando a remoção de ruído. Dentre as técnicas existentes na literatura, estão as redes neurais *autoencoders*. Nesse cenário, duas redes neurais são usadas conjuntamente para solucionar o problema de classificação de imagens ruidosas. Considerando isso, a pesquisa questiona o uso de informações do modelo de classificação para desenvolver e adaptar o modelo de remoção de ruído. Sendo assim, o objetivo geral é explorar um possível incremento na acurácia de um classificador de imagens usando um removedor de ruído *autoencoder*, adaptando-o pela introdução da métrica de erro da classificação em seu algoritmo de aprendizagem. O escopo do trabalho está limitado ao problema de classificação de imagens de dígitos manuscritos — escolhido pela sua relevância e complexidade de estudo tangível. O método utilizado consiste em uma pesquisa experimental composta por quatro procedimentos, sendo eles: (I) analisar o desempenho de uma rede neural em resolver o problema de classificação de dígitos em imagens sem ruído; (II) avaliar o desempenho do classificador, quando utilizado em imagens ruidosas; (III) estudar o impacto no desempenho de classificação ocasionado pelo pré-processamento das imagens; (IV) explorar a adaptação ao modelo de remoção de ruído e analisar seu impacto no problema de classificação estudado. Dentre os resultados, destaca-se que a adaptação explorada pode aprimorar a taxa de classificações corretas para imagens ruidosas pré-processadas. Além disso, são observados indícios que a adaptação torna o desenvolvimento dos modelos de remoção de ruído mais eficientes. Para trabalhos futuros é citado o desenvolvimento de uma pesquisa explicativa objetivando detalhar formalmente os resultados obtidos. Outra possibilidade é replicar a adaptação explorada em outros escopos que fazem o uso conjunto de duas (ou mais) redes neurais.

Palavras-chave: Aprendizado Profundo; Classificação de Imagens; Remoção de Ruído.

ABSTRACT

Artificial Intelligence is the Computer Science field that seeks to understand and develop intelligent machines, capable of taking efficient actions to solve problems in unknown scenarios. Among its subfields is Deep Learning, represented by biological brain inspired algorithms for artificial neural networks development. Its relevance lies in the possibility of application to different problems in different areas. As an example, and the research focus, image classification is cited — characterized by the need to assign one (or more) label(s) to an image based on the information it represents. However, it is observed that neural networks are negatively affected by the noise presence in data. This impact can be reduced by applying pre-processing methods for noise removal. Among the existing techniques in the literature are the autoencoder neural networks. In this scenario, two neural networks are used together to solve the noisy image classification problem. Considering this, the research questions the use of information from the classification model to develop and adapt the noise removal model. Therefore, the general objective is to explore the possibility of an image classifier accuracy increase using an autoencoder noise remover, adapting it by introducing the classification error metric into its learning algorithm. The research scope is limited to the handwritten digits image classification problem — chosen for its relevance and tangible study complexity. The method used consists of experimental research consisting of four procedures, namely: (I) analyze the performance of a neural network in solving the digit classification problem in noise-free images; (II) evaluate the classifier performance when used on noisy images; (III) study the impact on classification performance caused by image pre-processing; (IV) explore the noise removal model adaptation and analyze its impact on the studied classification problem. Among the results, it is highlighted that the adaptation explored can improve the correct classifications rate for pre-processed noisy images. Furthermore, there is evidence that adaptation makes the development of noise removal models more efficient. For future work is mentioned an explanatory research development aiming to formally detail the results obtained. Another possibility is to replicate the adaptation explored in other scopes that make joint use of two (or more) neural networks.

Keywords: Artificial Intelligence; Image Classification; Noise Removal.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADALINE	*Adaptative Linear Neuron* – Neurônio Linear Adaptativo
ANN	*Artificial Neural Network* – Rede Neural Artificial
CV	*Computer Vision* – Visão Computacional
DL	*Deep Learning* – Aprendizado Profundo
IA	Inteligência Artificial
IEEE	*Institute of Electrical and Electronics Engineers* – Instituto de Engenheiros Eletricistas e Eletrônicos
LMS	*Least Mean Squares*
ML	*Machine Learning* – Aprendizado de Máquina
MLP	*Multilayer Perceptron* – Perceptron de Multicamadas
MNIST	*Modified National Institute of Standards and Technology*
MSE	*Mean Square Error* – Erro Quadrático Médio
MTL	*Multitask Learning* – Aprendizado Multitarefas
OCR	*Optical Character Recognition* – Reconhecimento Óptico de Caracteres
SLP	*Single Layer Perceptron* – Perceptron de Camada Única

SUMÁRIO

1 INTRODUÇÃO

A Inteligência Artificial (IA) é um importante campo de estudos multidisciplinar da Ciência da Computação. Tem como objetivo compreender e desenvolver máquinas inteligentes, capazes de tomar ações eficientes em cenários desconhecidos. É uma área ativa com diversos tópicos de pesquisa em aberto e uma variedade de aplicações práticas (NILSSON, 1980; WINSTON, 1992; GOODFELLOW; BENGIO; COURVILLE, 2016; RUSSELL; NORVIG, 2021).

A área compreende subcampos de estudos que abordam o tema de formas específicas. Dentre eles, destaca-se o *Machine Learning* – Aprendizado de Máquina (ML) cuja especificidade está no uso de dados para o reconhecimento de padrões e construção de modelos para diferentes problemas (GOODFELLOW; BENGIO; COURVILLE, 2016; RUSSELL; NORVIG, 2021). Outra área de pesquisa em IA é o *Deep Learning* – Aprendizado Profundo (DL), sendo um subcampo de ML.

Segundo Ekman (2021, p. 26, tradução nossa):

> [...] DL é uma classe de algoritmos de aprendizado de máquina que usa múltiplas camadas de unidades computacionais, sendo que cada camada aprende sua própria representação para os dados de entrada. Essas representações são combinadas por camadas posteriores em uma moda hierárquica.

Sendo assim, o DL busca construir modelos para resolução de problemas através da combinação de unidades simples de processamento, capazes de reconhecer padrões a partir de dados, em camadas. Devida a sua origem ser inspirada pelo de processamento e pelas conexões neurais do cérebro biológico, os modelos desenvolvidos são usualmente referenciados como *Artificial Neural Network* – Rede Neural Artificial (ANN)[1] (HAGAN *et al.*, 2014; GOODFELLOW; BENGIO; COURVILLE, 2016; RUSSELL; NORVIG, 2021).

Machine Learning e *Deep Learning* são temas de relevância para a área de IA e sua popularização nas últimas duas décadas. Têm sido aplicados em diferentes indústrias como a automobilística, agrícola, financeira, médica, etc. (HAGAN *et al.*, 2014; SHINDE; SHAH, 2018; SARKER, 2021). Isso ocorre, pois demonstram ser alternativas aos métodos manuais para resolução de problemas e importantes para o desenvolvimento de teorias da computação (BISHOP, 2006; LECUN; BENGIO; HINTON, 2015; GOODFELLOW; BENGIO; COURVILLE, 2016; RUSSELL; NORVIG, 2021).

Dentre os campos da Ciência da Computação beneficiados por ML e DL, destaca-se a Visão Computacional — em inglês, *Computer Vision* (CV). Essa área de estudo, segundo Prince (2012), tem como objetivo manipular e extrair informações de imagens — uma tarefa não trivial. Para o autor o desenvolvimento do *Machine Learning* tem impacto direto no avanço da Visão Computacional, conforme explicitado em seu trabalho:

> Outra razão para o progresso recente nesta área (Visão Computacional) tem sido o aumento do uso do aprendizado de máquina. Nos últimos 20 anos vimos desenvolvi-

[1] É comum na literatura da área de IA usar o termo Rede Neural (ou *Neural Network*) como sinônimo de Redes Neurais Artificiais.

> mentos emocionantes neste campo de pesquisa em paralelo, e estes agora estão implantados amplamente em aplicações de visão. O aprendizado de máquina não somente fornece muitas ferramentas úteis, mas também nos ajudou a entender os algoritmos existentes e suas conexões sob uma nova perspectiva (PRINCE, 2012, p. 2, tradução nossa).

Como exemplo onde são aplicadas técnicas de ML e DL em CV, cita-se a classificação de imagens. Ela é caracterizada pela necessidade de atribuir um (ou mais) rótulo(s) a uma imagem de acordo com a informação que ela representa. Ela demonstra sua relevância pelo potencial de aplicação para resolução de problemas e pelo contínuo interesse no desenvolvimento de métodos eficientes (SZELISKI, 2010; FORSYTH; PONCE, 2011; RUSSELL; NORVIG, 2021).

Apesar dos avanços recentes no tópico de estudo mencionado, existem situações que dificultam a classificação de imagens usando ML e DL. Como destaque, e foco do presente trabalho, é mencionada a presença de ruído nos dados aplicados ao modelo de rede neural. Nesse contexto define-se ruído como uma perturbação aplicada ao dado para alterar o seu valor padrão (SZELISKI, 2010).

Segundo Goodfellow, Bengio e Courville (2016, p. 234, tradução nossa) "As redes neurais provam não serem muito robustas a ruídos [...]". Essa afirmação é justificada com resultados empíricos, como os destacados em Dodge e Karam (2016), Zhou, Song e Cheung (2017), Nazaré *et al.* (2018) e Momeny *et al.* (2021). Sendo assim, é necessária a adoção de técnicas para tornar as soluções robustas a essas perturbações nos dados.

Uma maneira de mitigar o impacto negativo de ruídos é modelar as soluções com dados que já possuem algum nível de perturbação (GOODFELLOW; BENGIO; COURVILLE, 2016). Todavia, para isso faz-se necessário acessar e modificar a rede neural. Outra alternativa compreende o uso de técnicas de pré-processamento de imagens para reduzir o ruído em imagens, possibilitando seu uso em um modelo previamente desenvolvido (NAZARÉ *et al.*, 2018).

Essa opção é interessante, observada a possibilidade de encontrar sistemas de classificação previamente modelados com dados sem perturbações (DODGE; KARAM, 2016). Dessa forma, é possível utilizar abordagens de remoção de ruído para padronizar os dados para que sejam similares aos utilizados pelo classificador. Assim, não há necessidade de modificar a solução original para obter robustez aos dados ruidosos.

Existem diversos algoritmos para remoção de ruído, reconhecida a relevância do tema para o processamento de imagens digitais (BUADES; COLL; MOREL, 2005). Dentre eles, é destacado o crescente desenvolvimento de técnicas baseadas em DL (GU; TIMOFTE, 2019), em especial, os modelos baseados em *autoencoders*[2]. Seu destaque é dado por serem técnicas robustas, que apresentam diferenciais em relação aos outros métodos (VINCENT *et al.*, 2008, 2010).

Portanto, dada a existência de uma rede neural para resolver um problema de classificação de imagens, é desejado adaptar a solução para torná-la robusta à presença de ruído nas imagens. Entretanto, não é possível remodelar o classificador com os dados ruidosos. Sendo assim, é

[2] Um tipo de rede neural artificial. Sua abordagem é feita durante o desenvolvimento do trabalho.

optado pelo uso de uma rede neural *autoencoder* para remoção de ruído durante etapa de pré-processamento.

Considerando que tanto a solução para classificação de imagens, quanto a solução para remoção de ruído são baseadas em redes neurais, é possível destacar um ponto em comum entre elas. Ambos os modelos dependem de uma medida — usualmente de erro — para que seus parâmetros possam ser atualizados durante a etapa de treino. Reduzir esse valor implica em melhorar o modelo, fazendo-o aprender a resolver o problema (HAGAN *et al.*, 2014; GOODFELLOW; BENGIO; COURVILLE, 2016).

Sendo assim, alterar o removedor de ruído afeta o desempenho do classificador de imagens. As alterações podem tanto aprimorar e reduzir o erro do modelo de classificação, quanto piorar e aumentar o seu erro. Logo, a ideia base para a elaboração do trabalho é explicitada no seguinte questionamento: é possível utilizar a medida de erro do classificador para adaptar o removedor de ruído visando uma maior taxa de classificações corretas?

Por fim, destaca-se a delimitação do escopo da pesquisa para a classificação de dígitos manuscritos. Conforme definido posteriormente, a natureza da pesquisa é de exploração, e a classificação de imagens é um tema amplo, logo, abordá-la de forma genérica não é trivial. Com a centralização dos esforços é permitido um estudo inicial que pode ser usado para fundamentar outros trabalhos.

Esse problema de classificação é escolhido especificamente, pois além de ser aplicado para soluções práticas em *Optical Character Recognition* – Reconhecimento Óptico de Caracteres (OCR), existem diversas pesquisas relacionadas que podem ser utilizadas como referência (LIU *et al.*, 2003; BALDOMINOS; SAEZ; ISASI, 2019). Além disso, é um problema equilibrado para a realização de um estudo acadêmico, pois apesar de ser uma tarefa não trivial, ele não exige um tremendo poder computacional para elaboração de uma solução (NIELSEN, 2015).

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Considerando o problema de classificação de dígitos manuscritos em imagens ruidosas, o objetivo geral da pesquisa é explorar a possibilidade de incrementar a acurácia de um modelo de classificação de imagens — treinado com imagens sem perturbações — por meio do uso e da adaptação de uma rede *autoencoder* para remoção de ruído. Detalhando, a adaptação é dada pela introdução do erro da classificação no processo de treino do modelo removedor de ruído.

1.1.2 Objetivos Específicos

Como objetivos específicos são listados:

- Identificar o nível de ruído que mais afeta negativamente a acurácia do classificador de imagens;
- Mensurar o maior ganho em acurácia ocasionado pela aplicação da remoção de ruído para o pré-processamento das imagens usadas pelo modelo de classificação;
- Dado o mecanismo para a adaptação do modelo de remoção de ruído, verificar o impacto ocasionado pela variação de seus parâmetros.

1.2 METODOLOGIA

A presente seção visa apresentar e justificar a metodologia adotada para a realização do trabalho. Primeiramente, uma classificação quanto aos tipos de pesquisa científica é mencionada para então apresentar o processo metodológico adotado e suas justificações. Ressalta-se que a seção em questão é feita sob uma visão geral, isto é, são abstraídas as complexidades teóricas dos procedimentos adotados — o detalhamento é feito na Seção 2.2.

Seguindo a caracterização apresentada por Wazlawick (2020) sobre os tipos de pesquisa, é destacada em sequência a classificação do trabalho em relação à sua natureza, aos seus objetivos e aos procedimentos técnicos adotados:

- **Quanto à natureza:** apesar do trabalho ser enquadrado sob áreas de estudo já estudadas, a formulação do problema e a abordagem proposta apresentam diferenciais em relação aos trabalhos relacionados. Portanto, a pesquisa é destacada principalmente como primária, caracterizada pelo estudo de um conhecimento pouco explorado;
- **Quanto aos objetivos:** no presente trabalho é buscado explorar uma abordagem específica para a resolução de um problema de classificação de imagens. Assim, são esperados resultados e discussões iniciais que possam ser usados para fundamentar estudos posteriores. Logo, a pesquisa é identificada essencialmente como exploratória;
- **Quanto aos procedimentos:** além da realização de uma pesquisa **bibliográfica** para revisão sistemática da literatura em artigos e livros objetivando a busca de conceitos e discussões para fundamentar o trabalho, é possível destacar a aplicação do procedimento **experimental**. No trabalho em questão, um conjunto de procedimentos são realizados manipulando variáveis relacionadas ao problema de classificação para produzir resultados refletidos sob uma métrica de análise — a acurácia.

No que diz respeito à pesquisa bibliográfica, os artigos foram selecionados com o auxílio de motores de busca como *Institute of Electrical and Electronics Engineers* – Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE), Google Scholar, Scielo, ScienceDirect e Scopus. Os termos utilizados correspondem ao tema da pesquisa e seu escopo. A filtragem das buscas é feita dando preferência para artigos da última década e pela quantidade de citações. Os livros

foram escolhidos com base nas referências utilizadas durante o curso de graduação, bem como outras de posse do autor.

Referente à pesquisa experimental, ela está baseada na realização e análise dos resultados obtidos em quatro experimentos, sendo eles:

- **Experimento I:** analisar o desempenho de uma rede neural — treinado em imagens sem perturbações — para resolver o problema de classificação de imagens de dígitos manuscritos. Dessa forma, é esperado mensurar a capacidade do modelo em resolver o problema e utilizar os resultados para o desenvolvimento dos experimentos seguintes;
- **Experimento II:** dado o classificador de imagens do experimento I, deseja-se analisar o seu desempenho quando utilizado para classificar imagens ruidosas. A intenção desse experimento é identificar e quantificar o impacto gerado pela adição de ruído na classificação de imagens;
- **Experimento III:** dado removedores de ruído, um classificador de imagens e um conjunto de imagens ruidosas, é buscado analisar o desempenho do classificador quando aplicado em imagens do conjunto de dados pré-processado pelo removedor de ruído. O propósito deste experimento é mensurar uma possível melhora na classificação após a etapa de pré-processamento;
- **Experimento IV:** apresentados diferentes modelos para remoção de ruído adaptados a partir do experimento anterior, deseja-se identificar se alguma das modificações aprimora o desempenho obtido na classificação de imagens. Esse experimento explora o objetivo principal da pesquisa.

Para a realização dos experimentos é necessário implementar duas redes neurais — um classificador de imagens e um removedor de ruído. Ambos são baseados no modelo *Multilayer Perceptron* – Perceptron de Multicamadas (MLP). As justificativas para sua escolha são: (I) É considerado um dos modelos mais importantes de DL (GOODFELLOW; BENGIO; COURVILLE, 2016); (II) É comprovada a sua aplicabilidade em problemas de classificação, incluindo o da pesquisa, bem como o seu uso no desenvolvimento de redes *autoencoders* (EKMAN, 2021).

Para treinar os modelos são necessários dados que representam o problema estudado. Considerando o escopo da pesquisa, foi optado pelo uso do conjunto de dados *Modified National Institute of Standards and Technology* (MNIST) de Bottou *et al.* (1994). Ele é formado por 70 mil imagens em escala de cinza de dígitos manuscritos (de 0 a 9). A sua escolha é feita pela disponibilidade pública dos dados[3] e pelo seu uso ser disseminado na área de IA.

Além dos dados, é necessária a adoção de um modelo de ruído para gerar as perturbações nas imagens. Com esse objetivo, faz-se o uso do modelo de Ruído Gaussiano Estacionário Aditivo. Sua escolha é defendida pelos argumentos: (I) É considerado um dos modelos de ruído mais comum (FORSYTH; PONCE, 2011), o que implica no seu uso em diferentes pesquisas;

[3] O conjunto original pode ser acessado diretamente em: <http://yann.lecun.com/exdb/mnist/>.

(II) Ruído gaussiano simula situações de corrupção reais causadas por má iluminação, alta temperatura e ruído de circuito eletrônico (CATTIN, 2016).

Dando sequência, destaca-se a métrica de avaliação — a acurácia — usada para expressar e analisar os resultados da pesquisa. Sua escolha é ocasionada, pois segundo Goodfellow, Bengio e Courville (2016) ela é comumente usada em problemas de classificação. Além disso, em comparação com outras medidas é dita ser mais intuitiva (EKMAN, 2021). Logo, para os fins do trabalho essa mensuração é suficiente para exploração do objetivo geral.

Por fim, para realizar os experimentos e desenvolver os códigos[4], é usada a linguagem de programação Python (3.11.5). Ela é optada pela familiaridade do autor com a linguagem, pela sua popularidade como ferramenta[5] e pela disponibilidade de referências (tutoriais, cursos e livros didáticos) para consulta. Além disso, é possível citar as diversas bibliotecas que agregam funcionalidades, facilitando a implementação de programas complexos.

Expandindo o tema sobre bibliotecas, é interessante destacar que a linguagem Python suporta recursos próprios para ML e DL, como PyTorch e TensorFlow. Todavia, na realização da pesquisa foi optado por não utilizá-los. A fundamentação dessa escolha está principalmente na necessidade de adaptar uma MLP para o Experimento IV, que é facilitada via implementação própria do modelo, uma vez que elimina restrições e padrões impostos pelas bibliotecas.

1.3 JUSTIFICATIVAS E CONTRIBUIÇÕES

As justificativas para a escolha do tema da pesquisa já estão listadas ao longo da introdução do presente trabalho. Recapitulando brevemente é possível destacar a relevância da área de Inteligência Artificial — em especial *Machine Learning* e *Deep Learning* — com tópicos ativos de pesquisa e diversas aplicações práticas. Além disso, problemas de classificação de imagens em Visão Computacional são um tópico de estudo importante.

Além disso, os resultados obtidos com a pesquisa têm potencial de aplicação prática. Conforme já citado, a presença de ruído afeta negativamente o desempenho de um classificador não robusto, logo, usar alternativas para contornar esse problema impacta na eficiência de sistemas. Como destaque o mecanismo estudado permite reaproveitar um modelo de classificação pré-existente, eliminando os gastos necessários para o seu re-desenvolvimento.

Outros argumentos relacionados ao método científico da pesquisa podem ser apresentados também. São eles: (I) Fácil replicação — o método e as ferramentas são de fácil compreensão e aplicação; (II) Diferencial em relação a outros trabalhos — conforme observado na Seção 1.4, os problemas de classificação de imagens e remoção de ruído são abordados em conjunto sob uma perspectiva diferente; (III) Possibilidade de trabalhos futuros — conforme menção na delimitação do escopo da pesquisa, os resultados obtidos podem ser replicados e utilizados no desenvolvimento de pesquisas na área.

[4] Disponíveis em: <https://github.com/Lima001/BCC-TCC>.

[5] Em novembro, a linguagem figura como a mais popular segundo dados do índice TIOBE (2023). Disponível em: <https://www.tiobe.com/tiobe-index/>. Acesso em: 04 nov. 2023.

Por fim, cita-se a contribuição principal da pesquisa que está na exploração inicial de um conjunto de ideias pela conciliação de teoria e prática. O propósito é reunir discussões e resultados que possam ser trabalhados não somente no contexto de visão computacional e de classificação de imagens, mas também serem estendidos para outras áreas e problemas que fazem o uso de conceitos de *Deep Learning*.

1.4 TRABALHOS CORRELATOS

Para os trabalhos correlatos são destacadas pesquisas cuja abordagem principal está relacionada com pelo menos um dos seguintes temas: (I) Classificação de imagens de dígitos manuscritos; (II) Classificação de imagens na presença de dados ruidosos; (III) Treino de redes neurais em conjunto para solucionar problemas relacionados. Essa limitação é feita, pois esses são os tópicos explorados ao longo do trabalho em questão.

Primeiramente, no que diz respeito a classificação de imagens de dígitos manuscritos, já foram citadas as pesquisas de Liu *et al.* (2003) e Baldominos, Saez e Isasi (2019). Ambos os trabalhos realizam uma revisão da literatura sobre o tema e podem ser usados para indicar trabalhos relacionados. Com base nos levantamentos apontados pelos autores, destaca-se a existência de diferentes abordagens para resolver o problema de classificação de dígitos.

Além de métodos manuais, são mencionadas pesquisas que fazem uso de redes neurais para solucionar o problema. Os principais resultados explicitam o emprego de técnicas de *Machine Learning* e *Deep Learning* para construir modelos eficientes — que realizam classificações com taxas superiores à 98% de sucesso. Sendo assim, com essas pesquisas é possível reafirmar a solubilidade do problema estudado.

Ainda sobre o tema de classificação, em Bottou *et al.* (1994) é possível notar uma pesquisa comparativa avaliando o desempenho de diferentes algoritmos com esse propósito. Além da apresentação e uso do conjunto de dados MNIST, é destacada a aplicação de uma rede neural MLP — mesmo método proposto na presente pesquisa — para solucionar o problema. Dentre os resultados relevantes, cita-se novamente a obtenção de altas taxas de classificações corretas.

Dando sequência, é possível explicitar que diversas pesquisas na área buscam desenvolver, comparar e analisar métodos para resolver o problema de classificação de dígitos. Porém, o objetivo desses trabalhos é primariamente a busca em obter o classificador com melhor desempenho. Logo, eles muitas vezes desconsideram situações adversas como a presença de ruído nos dados.

Sendo assim, diferentemente das pesquisas destacadas pelos autores supramencionados, o trabalho em questão não está preocupado em obter o melhor desempenho possível. O objetivo explorado está relacionado ao contexto adverso de classificação, elaborado a partir da degradação das imagens a serem classificadas pela introdução de ruído nos dados. Observada essa questão, em sequência são mencionados trabalhos relacionados que estudam cenários similares.

Como destaque são apresentados os trabalhos de Zhou, Song e Cheung (2017) e Momeny *et al.* (2021). Ambos abordam de alguma forma o problema de classificação de dígitos provenientes do MNIST e consideram a introdução de ruído gaussiano nas imagens. Dentre os resultados e conclusões obtidas pelas pesquisas, está a observação negativa do desempenho dos algoritmos de classificação quando aplicados em dados ruidosos.

Para solucionar o problema ocasionado pelas perturbações nos dados e obter melhores taxas de classificações corretas, cada pesquisa apresenta propostas de adaptações nos modelos de classificação de imagens desenvolvidos. Sendo assim, uma diferença para o trabalho proposto está na restrição de não poder modificar o modelo de classificação, forçando o uso de um método para remoção de ruído.

Sobre o uso de métodos de remoção de ruído, em especial soluções baseadas em redes *autoencoders*, para a classificação de imagens ruidosas de dígitos é possível citar trabalhos de Roy, Ahmed e Akhand (2017, 2018). Dentre os resultados, são obtidas melhores taxas de classificação — inclusive para o conjunto de dados MNIST — com a aplicação do pré-processamento dos dados usando redes neurais.

Todavia, nenhum dos trabalhos até então mencionados, bem como os discutidos por eles, estipulam um escopo similar ao proposto pela pesquisa atual. Conforme o objetivo geral da pesquisa, é desejado explorar a introdução de uma informação (erro) proveniente do modelo para classificação no processo de treino do modelo para remoção de ruído. Assim, espera-se identificar possíveis variações no desempenho geral de classificação.

A título de explicitação, ideia similar é encontrada na área de *Multitask Learning* – Aprendizado Multitarefas (MTL). Nela são buscadas desenvolver conjuntamente soluções de ML — incluindo DL — para problemas relacionados através do compartilhamento de informações entre modelos. Pesquisas como as de Li *et al.* (2017) e Xu *et al.* (2020) exemplificam a eficiência dessa abordagem para classificação de imagens, porém usando técnicas e escopos diferentes do trabalho em questão.

Para finalizar a seção, o Quadro 1 resume os trabalhos correlatos por abordagem e destaca o diferencial do trabalho. É perceptível a existência de pesquisas que abordam o tema de classificação de dígitos manuscritos, inclusive para dados ruidosos. Todavia, a exploração proposta é uma característica do trabalho em questão, diferindo de pesquisas em MTL principalmente pelas técnicas e problemas estudados.

Quadro 1 – Trabalhos correlatos por abordagem.

Tema da pesquisa	**Trabalhos identificados**
Classificação de imagens em dígitos manuscritos.	Bottou *et al.* (1994); Liu *et al.* (2003); Baldominos, Saez e Isasi (2019).
Classificação de imagens em dígitos manuscritos na presença de dados ruidosos.	Roy, Ahmed e Akhand (2017) Zhou, Song e Cheung (2017); Nazaré *et al.* (2018); Roy, Ahmed e Akhand (2018); Momeny *et al.* (2021).
Treino de redes neurais em conjunto para solucionar problemas relacionados.	Li *et al.* (2017); Xu *et al.* (2020).
Adaptação de rede neural pelo uso conjunto de modelos para problemas relacionados.	Nenhuma pesquisa identificada.

Fonte: Elaboração própria.

2 DESENVOLVIMENTO

A presente seção apresenta o desenvolvimento da pesquisa. Primeiramente são destacados os aspectos teóricos que fundamentam o trabalho e são minimamente necessários para compreender os procedimentos experimentais desenvolvidos. Após isso, são detalhados individualmente os experimentos realizados para alcançar os objetivos do trabalho. Por fim, os resultados obtidos são explorados, e a finalização da pesquisa é iniciada.

2.1 FUNDAMENTAÇÃO TEÓRICA

A seção teórica do trabalho pode ser dividida na abordagem de dois temas principais — fundamentos de *Deep Learning* e conceitos de processamento de imagens. O primeiro busca destacar os conceitos necessários para compreender o funcionamento, desenvolvimento e aplicação dos modelos de redes neurais usados na pesquisa. Já o segundo apresenta tópicos básicos de representação de imagens, bem como o modelo de ruído utilizado no trabalho.

Antes de prosseguir, é importante explicitar que os tópicos, principalmente de DL, são abordados com o auxílio do formalismo matemático. Logo, é comum a apresentação de símbolos e equações durante o texto. Por isso destaca-se o uso de notação similar à observada em Hagan *et al.* (2014), sendo o padrão de identificação: Escalar a (minúsculo e itálico); Matriz $\mathbf{W}$ (maiúsculo e negrito); Vetor $\vec{\mathbf{u}}$ (minúsculo e negrito); Outros tipos de dados são previamente mencionados e identificados.

2.1.1 Redes Neurais Artificiais

Segundo Haykin (2008) uma rede neural artificial é um dispositivo que busca modelar a maneira que o cérebro humano executa determinada tarefa, ou função de interesse. É usualmente desenvolvida diretamente em hardware, ou simulada via software em um computador digital. A sua relevância está baseada na capacidade de desenvolver soluções para problemas complexos por meio de um processo adaptativo, similar ao que ocorre na aprendizagem humana.

A sua origem, em conformidade com Hagan *et al.* (2014), está intimamente relacionada com a tentativa de compreender a estrutura e funcionamento do cérebro biológico. Estudos iniciais datam do final do séc. XIX e início do séc. XX, sendo a maioria referente às pesquisas interdisciplinares em física, psicologia e neuropsicologia. Já o estudo computacional do tema começa por volta de 1940 com o trabalho de McCulloch e Pitts (1943)[6].

Apesar de sua origem, as redes neurais artificiais (por enquanto) não abordam as complexidades intrínsecas do cérebro biológico em sua totalidade. De acordo com Hagan *et al.* (2014) as similaridades entre o modelo artificial e o biológico são dadas por dois fatores chaves, sendo eles: (I) São constituídos de unidades de processamento simples — neurônios —, sendo elas

[6] Demonstrava que redes neurais poderiam (até então) calcular qualquer função aritmética ou lógica. Serviu como base para o desenvolvimento e avanço de teorias e aplicações na área de ML e DL (HAGAN *et al.*, 2014).

altamente interconectadas formando uma estrutura de rede; (II) A funcionalidade da rede neural é definida pelas conexões entre os seus neurônios.

Agregando a discussão sobre as similaridades entre os modelos biológico e artificial, Haykin (2008, p. 2, tradução nossa) em sua definição de rede neural artificial como uma máquina adaptativa cita:

> Uma rede neural é um processador distribuído massivamente paralelo composto de unidades simples de processamento que têm uma propensão natural para armazenar conhecimento experiencial e disponibilizá-lo para uso. Assemelha-se ao cérebro em dois aspectos: 1. O conhecimento é adquirido pela rede a partir do seu ambiente através de um processo de aprendizagem; 2. As forças de conexão entre neurônios, conhecidas como pesos sinápticos, são usadas para armazenar o conhecimento exigido.

Em suma as ANN's são modelos computacionais para processamento de dados formados pela composição de unidades simples interconectadas. A sua aplicabilidade está na capacidade de utilizar dados para adaptar os parâmetros do modelo em um processo chamado de aprendizagem. Dessa forma, é possível que a rede neural encontre iterativamente uma configuração capaz de solucionar o problema para o qual foi projetada.

Dando continuidade, para compreender um modelo de rede neural faz-se necessário abordar inicialmente as unidades simples de processamento que o compõem. Elas são chamadas de neurônios artificiais e consistem basicamente de um conjunto de operações matemáticas aplicadas sob uma entrada para a produção de uma saída. Em sequência é apresentado o modelo teórico de um neurônio e exemplificado o seu processamento de dados.

2.1.1.1 Modelo de Neurônio Artificial

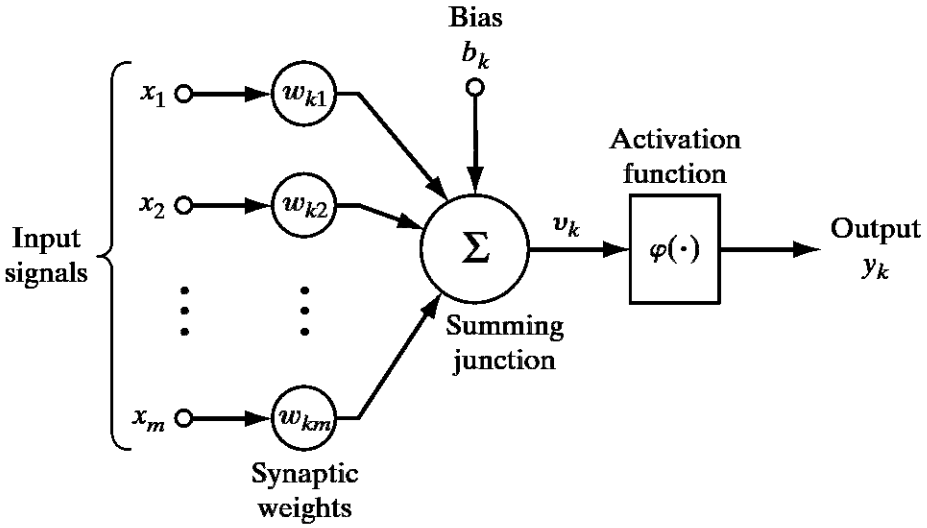
Com base em Haykin (2008) o modelo teórico para um neurônio artificial genérico k (Figura 1) consiste de[7]:

1. uma unidade de processamento formada por um somador (*junction summing*) $\sum$ e uma função matemática genérica $\varphi : \mathbb{C} \to \mathbb{C}$ chamada de função de ativação (*activation function*);
2. sinais de entrada (*input signals*) $x_0, x_1, \ldots, x_m$, sendo $x_0 = 1$ um sinal fixo;
3. pesos sinápticos (*synaptic weights*) $w_{k0}, w_{k1}, \ldots, w_{km}$, sendo $w_{k0} = 1$ um peso especial denominado *bias*. Detalhando a notação, cada peso do neurônio está associado a um sinal de entrada, sendo assim o primeiro e o segundo índice identificam respectivamente o neurônio k, e o sinal de entrada i associado ao peso;
4. o potencial de ativação (*activation potential*) v_k, que representa o valor processado pelo somador;

[7] O padrão de nomenclatura segue Haykin (2008), todavia outros nomes são encontrados na literatura. Os mais comuns são respectivamente: 1. *summer*; 2. *squashing function* e *transfer function*; 3. *offset*; 4. *strenghts*; 5. *induced local field* e *net input*.

5. a saída do neurônio (*output*) y_k, frequentemente normalizada entre os intervalos $[0,1]$ ou $[-1,1]$.

Figura 1 – Modelo teórico de neurônio artificial.



Fonte: Adaptado de Haykin (2008, p. 11)

Analisando o modelo do neurônio artificial é possível observar suas similaridades com o neurônio biológico. Segundo Ekman (2021, paginação irregular, tradução nossa):

> Um neurônio biológico consiste em um corpo celular, vários dendritos e um único axônio. As conexões entre neurônios são conhecidas como sinapses. O neurônio recebe estímulos nos dendritos e em casos de estímulos suficientes, o neurônio dispara (também conhecido como processo de ativação ou excitação) e emite estímulo em seu axônio, que é transmitido para outros neurônios que possuem conexões sinápticas com o neurônio excitado. Os sinais sinápticos podem ser excitatórios ou inibitórios; isto é, alguns sinais podem impedir que um neurônio dispare em vez de fazê-lo disparar.

O modelo artificial também possui um corpo — unidade de processamento — que recebe os estímulos — sinais de entrada — usados para gerar um disparo — saída. Além disso, a excitação ou inibição de sinais sinápticos é feita pelos pesos, que aumentam ou diminuem o valor do sinal. Por fim, destaca-se como diferença o *bias*, sendo ele um parâmetro externo presente no modelo artificial.

Considerado isso, em concordância com Haykin (2008) é possível definir um neurônio artificial k matematicamente pelas equações

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{1}$$

e

$$y_k = \varphi(v_k) \tag{2}$$

onde v_k consiste da soma dos produtos entre os sinais de entrada e seus respectivos pesos (incluindo o *bias*). Já y_k corresponde a saída do neurônio, obtida ao aplicar a função de ativação sobre o potencial de ativação.

Para agregar a discussão do tema, cabe uma breve explicação do parâmetro *bias* e sua relevância para o modelo. Matematicamente a Equação (2) é equivalente a

$$y_k = \varphi(u_k + b_k) \tag{3}$$

sendo u_k identificado como a saída do combinador linear, obtido pela equação

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{4}$$

Portanto, o propósito do *bias* é aumentar, ou diminuir o valor usado em φ — representa uma transformação afim —, flexibilizando o intervalo de valores de saída gerados pelo neurônio.

Ainda é comum encontrar na literatura a representação matricial do neurônio (HAYKIN, 2008; HAGAN *et al.*, 2014; EKMAN, 2021), dada por

$$\vec{\mathbf{x}} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{5}$$

um vetor formado pelos sinais de entrada,

$$\mathbf{W} = \begin{bmatrix} w_{k0} & w_{k1} & w_{k2} & \dots & w_{km} \end{bmatrix} \tag{6}$$

a matriz de pesos. Dessa forma, o potencial de ativação

$$\vec{\mathbf{v}}_\mathbf{k} = \mathbf{W}\vec{\mathbf{x}} \tag{7}$$

é reescrito como o vetor resultante da transformação da matriz de pesos sobre o vetor de sinais de entrada. Com isso, para a obtenção da saída da rede

$$\vec{\mathbf{y}}_\mathbf{k} = \varphi(\vec{\mathbf{v}}_\mathbf{k}) \tag{8}$$

adapta-se a definição da função de ativação para aplicá-la sob um vetor. Sendo assim, para o vetor unidimensional $\vec{\mathbf{v}}_\mathbf{k}$ define-se que

$$\varphi(\vec{\mathbf{v}}_\mathbf{k}) = \Big(\varphi(v_1)\Big) \tag{9}$$

isto é, a operação da função de ativação deve ser aplicada à primeira (e única) componente do vetor representando o potencial de ativação da rede — $\vec{\mathbf{v}}_\mathbf{k}$.

Fazendo proveito da menção à função de ativação, é importante destacar que existem diferentes tipos de funções (lineares ou não) que são usadas. A escolha de uma cabe ao projeto do modelo de rede neural e deve levar em conta aspectos do problema a ser resolvido (HAGAN *et al.*, 2014). Exemplos são observados no trabalho do autor mencionado e, quando necessários, explicitados na pesquisa.

2.1.1.2 Arquiteturas de Redes Neurais

O potencial das ANN's está na composição dos neurônios para formar modelos complexos. Conforme observado em Haykin (2008) e Hagan *et al.* (2014) são identificadas três arquiteturas, modos de organizar os neurônios, fundamentais de redes neurais. São elas as Redes *Feedforward* de Camada Única, as Redes *Feedforward* Multicamadas e as Redes Recorrentes[8].

As Redes *Feedforward* de Camada Única, ou *Single-Layer Feedforward Networks* são modelos onde um, ou mais neurônios são agrupados paralelamente e sem conexão entre si. Esse agrupamento recebe o nome de camada. Do ponto de vista formal, dada uma rede *Single-Layer Feedforward* genérica de n neurônios (Figura 2), o processamento do seu vetor de saída

$$\vec{\mathbf{y}} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \tag{10}$$

é definido pela equação $\vec{\mathbf{y}} = \varphi(\vec{\mathbf{v}}) = \varphi(\mathbf{W}\vec{\mathbf{x}})$, sendo

$$\vec{\mathbf{x}} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{11}$$

um vetor formado pelos m sinais de entrada,

$$\mathbf{W} = \begin{bmatrix} w_{10} & w_{11} & w_{12} & \cdots & w_{1m} \\ w_{20} & w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{n0} & w_{n1} & w_{n2} & \cdots & w_{nm} \end{bmatrix} \tag{12}$$

a matriz composta pelos pesos de todos os n neurônios,

$$\vec{\mathbf{v}} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \tag{13}$$

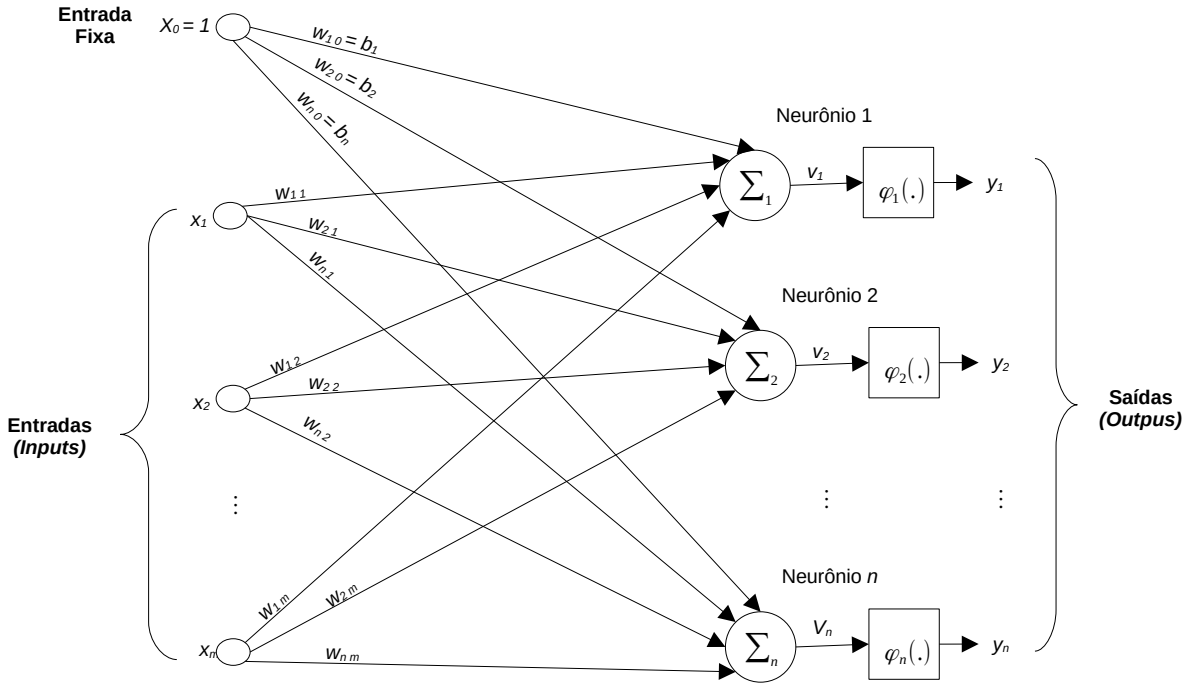
formado pelo potencial de ativação dos n neurônios da camada e

$$\varphi(\vec{\mathbf{v}}) = \begin{pmatrix} \varphi_1(v_1) \\ \varphi_2(v_2) \\ \vdots \\ \varphi_n(v_n) \end{pmatrix} \tag{14}$$

[8] Considerando o escopo da pesquisa, esse tipo de arquitetura não é abordado na discussão teórica.

indicando a aplicação da função de ativação de cada neurônio ao seu respectivo potencial de ativação.

Figura 2 – Modelo teórico de rede neural *Single-Layer Feedforward*.



Fonte: Elaboração própria

Especificando a notação da Equação (12) para a matriz **W**, seja w_{ij} um de seus elementos, o índice *i* corresponde ao identificador do neurônio ao qual o peso é referente, e *j* é o identificador que associa o peso à entrada de mesmo índice. Portanto, w_{21} é o peso associado ao sinal de entrada x_1 para o neurônio 2. Dessa forma, cada linha da matriz representa individualmente o conjunto de pesos de um único neurônio.

Já no que diz respeito a notação da Equação (14), ela é uma adaptação da representação observada anteriormente na Equação (9). Sabendo que $\vec{\mathbf{v}}$ é o vetor que representa o potencial de ativação de todos os neurônios da rede, faz-se necessário que a função de ativação de cada neurônio seja aplicada ao seu respectivo valor nesse vetor. Sendo assim, $\varphi(v_i)$ implica na aplicação da função do neurônio *i* ao seu potencial de ativação identificado respectivamente pela componente *i* do vetor $\vec{\mathbf{v}}$.

Prosseguindo, cabe ressaltar alguns pontos sobre essa arquitetura: (I) A quantidade de sinais de entradas e neurônios são escolhas de projeto, sendo limitada apenas a quantidade de camadas; (II) A saída da rede será um vetor *n*-dimensional, onde *n* é o número de neurônios; (III) Cada neurônio processa a entrada conforme o modelo teórico da Seção 2.1.1.1; (IV) A terminologia *Feedforward* refere-se ao processamento unidirecional da rede, da entrada para a saída.

Dando sequência, é apresentada a arquitetura de Redes *Feedforward* de Multicamada, ou *Multilayer Feedforward Networks*. A sua diferença em relação a anterior é a adição de uma,

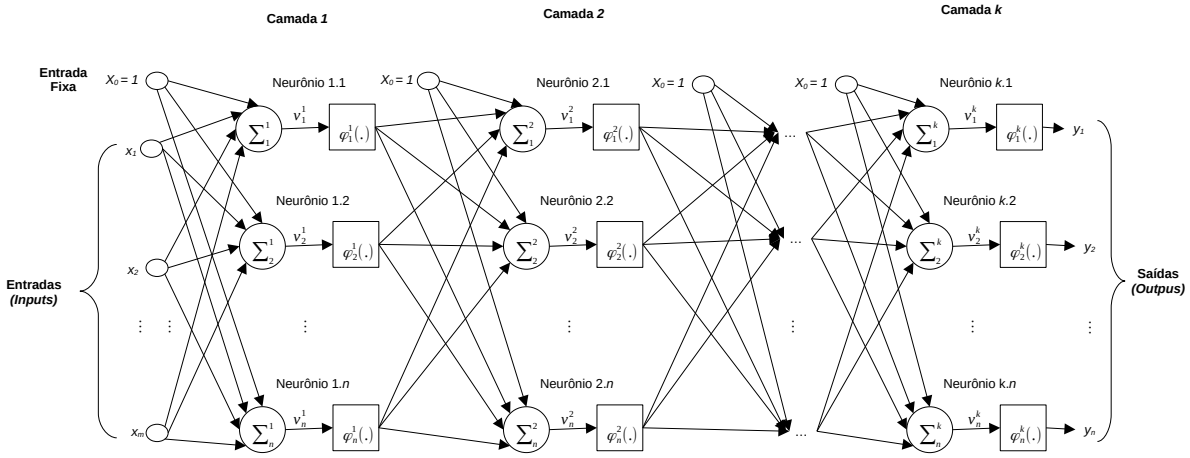
ou mais camadas intermediárias (ocultas) entre os sinais de entrada e os neurônios de saída da rede. Com isso, a partir dos sinais de entrada, os valores processados pelos neurônios de uma camada são encaminhados aos neurônios da próxima camada até que sejam obtidos os valores finais de saída.

Matematicamente a formulação de uma rede *Multilayer Feedforward* genérica de k camadas (Figura 3) para o processamento do vetor de saída $\vec{\mathbf{y}}$ (igualmente definido pela Equação (10)) é dada pela composição de funções

$$\vec{\mathbf{y}} = \varphi^k(\mathbf{W^k}(\ldots(\varphi^2(\mathbf{W^2}(\varphi^1(\mathbf{W^1}\,\vec{\mathbf{x}}))))))) \tag{15}$$

sendo $\mathbf{W^i}$ as matrizes compostas pelos pesos de todos os n neurônios da camada i, $\vec{\mathbf{x}}$ os m sinais de entrada e $\varphi^i(\vec{\mathbf{v}})$ indicando a aplicação das funções de ativação dos neurônios específicos da camada i da rede neural.

Figura 3 – Modelo teórico de rede neural *Multilayer Feedforward*.



Fonte: Elaboração própria

A definição formal de $\vec{\mathbf{x}}$ ocorre da mesma forma que observada em (11). Entretanto, as notações para os pesos e as funções de ativação necessitam ser modificadas, uma vez que cada camada p de neurônios necessita individualmente de um conjunto de funções de ativação

$$\varphi^p(\vec{\mathbf{v}}) = \begin{pmatrix} \varphi_1^p(v_1) \\ \varphi_2^p(v_2) \\ \vdots \\ \varphi_n^p(v_n) \end{pmatrix} \tag{16}$$

e uma matriz de pesos para os neurônios

$$\mathbf{W^p} = \begin{bmatrix} w_{10}^p & w_{11}^p & w_{12}^p & \cdots & w_{1m}^p \\ w_{20}^p & w_{21}^p & w_{22}^p & \cdots & w_{2m}^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{n0}^p & w_{n1}^p & w_{n2}^p & \cdots & w_{nm}^p \end{bmatrix} \tag{17}$$

Detalhando as notações para a matriz $\mathbf{W^p}$, dado o elemento w_{ij}^p, os índices i e j são respectivamente o neurônio ao qual o peso está associado, e o valor de entrada referente. Já para $\varphi^p(\vec{\mathbf{v}})$ é usado o mesmo artifício para identificar as diferentes camadas, portanto, $\varphi_i^p(v_i)$ implica na aplicação da função de ativação do neurônio i localizado na camada p sob o seu respectivo potencial de ativação v_i.

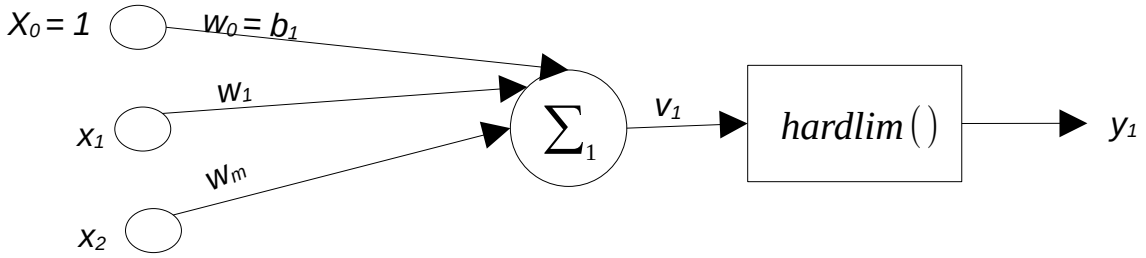
Aqui cabe fazer a ressalva de um aspecto da Figura 3, a notação e identificação individual dos neurônios. Para cada neurônio é atribuído um identificador composto de dois escalares sob a forma *i.j*, logo, *i* identifica a camada e *j* individualmente o neurônio. Como observação cita-se que as diferentes camadas podem ser formadas por quantidades arbitrárias de neurônios, que não precisam ser iguais entre elas.

Finalizando, é destacado que modelos desse tipo são computacionalmente mais poderosos que os anteriores. Conforme Haykin (2008) a adição de camadas permite extrair representações mais complexas dos dados. Segundo Hagan *et al.* (2014, paginação irregular, tradução nossa) "[...] uma rede de duas camadas [...] pode ser treinada para aproximar arbitrariamente bem a maioria das funções. As redes de camada única não podem fazer isso".

2.1.1.3 Processo de Aprendizagem

Cada neurônio de uma rede neural artificial possui um conjunto de pesos. Esses são também chamados de parâmetros ajustáveis do modelo e correspondem aos valores que devem ser obtidos pela ANN para produzir as saídas esperadas (ou mais próximas das esperadas), solucionando um problema (HAGAN *et al.*, 2014). Para fins de exemplo, apresenta-se uma rede neural (Figura 4) usada para elaboração do tema da seção.

Figura 4 – Exemplo de rede neural *Single-Layer* com um neurônio.



Fonte: Elaboração própria

Esse modelo recebe verdadeiramente apenas dois sinais de entrada $\vec{\mathbf{x}}^T = \begin{bmatrix} 1 & x_1 & x_2 \end{bmatrix}$ — o escalar fixo refere-se a adaptação para inclusão do *bias*. No que diz respeito a sua função de ativação, ela é chamada de *hard limit* (*hardlim*) e é defina por

$$hardlim(n) = \begin{cases} 0, & n < 0 \\ 1, & n \geq 0 \end{cases} \tag{18}$$

Considerando as matrizes de pesos $\mathbf{W_a} = \begin{bmatrix} -1{,}5 & 1 & 1 \end{bmatrix}$ e $\mathbf{W_b} = \begin{bmatrix} -0{,}5 & 1 & 1 \end{bmatrix}$ o Quadro 2

mostra o uso delas sob um mesmo conjunto de sinais de entrada e exibe as saídas processadas pelas redes.

Quadro 2 – Mapeamento de sinais de entrada para as saídas de um neurônio (para os pesos $\mathbf{W_a}$ e $\mathbf{W_b}$).

Sinais de Entrada	Saída para Neurônio com pesos Wa	Saída para Neurônio com pesos Wb
$\vec{\mathbf{x_1}} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$	$y_k = hardlim(\mathbf{W_a}\vec{\mathbf{x_1}}) = 0$	$y_k = hardlim(\mathbf{W_b}\vec{\mathbf{x_1}}) = 0$
$\vec{\mathbf{x_2}} = \begin{bmatrix}1 & 0 & 1\end{bmatrix}^T$	$y_k = hardlim(\mathbf{W_a}\vec{\mathbf{x_2}}) = 0$	$y_k = hardlim(\mathbf{W_b}\vec{\mathbf{x_2}}) = 1$
$\vec{\mathbf{x_3}} = \begin{bmatrix}1 & 1 & 0\end{bmatrix}^T$	$y_k = hardlim(\mathbf{W_a}\vec{\mathbf{x_3}}) = 0$	$y_k = hardlim(\mathbf{W_b}\vec{\mathbf{x_3}}) = 1$
$\vec{\mathbf{x_4}} = \begin{bmatrix}1 & 1 & 1\end{bmatrix}^T$	$y_k = hardlim(\mathbf{W_a}\vec{\mathbf{x_4}}) = 1$	$y_k = hardlim(\mathbf{W_b}\vec{\mathbf{x_4}}) = 1$

Fonte: Elaboração própria.

Especificando, o modelo apresentado é conhecido como *Perceptron* de Rosenblatt (discutido na Seção 2.1.1.5). Além disso, os pesos $\mathbf{W_a}$ e $\mathbf{W_b}$ formam redes neurais que mapeiam respectivamente as funções *AND* e *OR* da lógica booleana. Por fim, é importante citar que o exemplo foi adaptado de discussões e exercícios em Hagan *et al.* (2014).

O quadro anterior é usado para explicitar a necessidade de saber os pesos corretos para obter um modelo que soluciona o problema desejado. Para situações simples, como a apresentada, é possível descobrir a matriz de pesos por meio de cálculos manuais, enquanto para problemas complexos — uma rede *Multilayer* com diversas entradas, camadas e neurônios — essa tarefa não é trivial. Sendo assim, para isso existem os processos e as regras de aprendizagem.

Os processos de aprendizagem consistem de categorias de algoritmos (também chamados de regras de aprendizado, ou *learning rules*) usados para atualizar os parâmetros de uma rede neural durante o seu processo de treino. As seguintes classificações de aprendizagem são as mais comuns: (I) Aprendizado supervisionado; (II) Aprendizado não-supervisionado (HAYKIN, 2008; HAGAN *et al.*, 2014).

No aprendizado supervisionado (*supervised learning*), uma ANN é treinada a partir de um conjunto de dados composto por pares de entrada e saída. Assim, para cada entrada a saída ótima esperada é conhecida, e os parâmetros do modelo são ajustados para aproximar esse dado. Essa categoria é comum em tarefas de classificação e regressão, sendo as regras de aprendizado Delta e de aprendizado do *Perceptron* seus exemplos clássicos.

Já na categoria de aprendizado não-supervisionado (*unsupervised learning*), como o nome sugere, não existem mapeamentos de entrada e saída esperada. O ajuste é realizado por meio da otimização de uma medida de qualidade independente. Essa categoria é comumente aplicada em tarefas de agrupamento e redução de dimensionalidade de dados, tendo como exemplos de algoritmos as regras de aprendizado hebbiana e os algoritmos de aprendizado competitivo.

2.1.1.4 Generalização, Treino, Validação e Inferência

Considerados os temas anteriormente apresentados, julga-se importante apresentar outros conceitos que auxiliam no entendimento das redes neurais. Recapitulando, ANN's são modelos computacionais usados para iterativamente encontrar uma solução para um problema. Do ponto

de vista formal é possível relacionar os modelos com aproximações de uma função que soluciona o problema explorado (HAYKIN, 2008; HAGAN *et al.*, 2014).

Além disso, a adaptação do modelo é feita através dos algoritmos de aprendizagem que modificam os parâmetros da rede neural (geralmente iniciados com valores aleatórios) para que ela produza saídas condizentes. Esse processo de iterativamente aprimorar o modelo é denominado de etapa de treino. Nela é necessário o uso de dados de exemplo para que possam ser aplicados aos algoritmos de aprendizado e o modelo seja treinado.

O conjunto de dados com esse objetivo é chamado de conjunto de treino e o seu formato depende do problema e tipo de aprendizado aplicado. Portanto, os dados devem expressar a solução objetivada. Por exemplo, uma ANN para classificar dados nas categorias A ou B treinada com muitos exemplos da categoria A e poucos da B terá problemas em desempenhar a classificação correta de entradas do segundo tipo (JOHNSON; KHOSHGOFTAAR, 2019)[9].

O conjunto de dados de treino é importante, pois é a partir dele que espera-se obter a generalização do modelo. Esse conceito refere-se à capacidade da rede neural em identificar os padrões e características dos dados de modo que seja capaz de solucionar o problema alvo para entradas desconhecidas (EKMAN, 2021). Dessa forma, sabe-se que o modelo não memorizou um conjunto de treino, mas sim tornou-se hábil a apresentar as soluções corretas.

Além do treino é comum realizar a fase de validação. Nela é simulada a aplicação do modelo em um contexto real com dados (pertencentes ao conjunto de validação) não usados durante o treino. Uma métrica de desempenho é usada para identificar o quão bem a generalização foi realizada e pode ser usada em comparação com os resultados obtidos durante a etapa de aprendizado, para identificar problemas[10], ou confirmar a eficiência da solução.

Após a finalização do desenvolvimento de um modelo de DL, ele pode ser utilizado em aplicações práticas. Nesse momento ele passa a ser utilizado como uma ferramenta de inferência (ou predição) — dadas entradas desconhecidas pelo modelo, se corretamente implementado ele deve produzir uma saída adequada ao contexto do problema (EKMAN, 2021). Por fim, destaca-se que generalização e inferência podem ser realizadas em hardwares distintos, uma vez que treinar a rede é computacionalmente mais custoso do que aplicá-la na prática.

Considerado isso, os conhecimentos apresentados até então são suficientes para entender a área de DL sob uma perspectiva geral. Todavia, para o desenvolvimento da pesquisa é necessário detalhar a fundamentação teórica para tratar dois exemplos de redes neurais. Por isso, as próximas seções apresentam, com certo rigor, os modelos *Perceptron* de Rosenblatt e *Multilayer Perceptron*.

---

[9] Esse problema é conhecido como desbalanceamento de classes (em inglês, *class imbalance*) e pode afetar negativamente o desempenho de redes neurais treinadas usando aprendizado supervisionado (JOHNSON; KHOSHGOFTAAR, 2019).

[10] Como exemplo clássico cita-se o *overfitting*. Caso o modelo apresente desempenho satisfatório na fase de treino, mas insatisfatório na validação, possivelmente a rede neural está memorizando os dados de treino e não generalizando o problema (EKMAN, 2021).

2.1.1.5 Perceptron de Rosenblatt

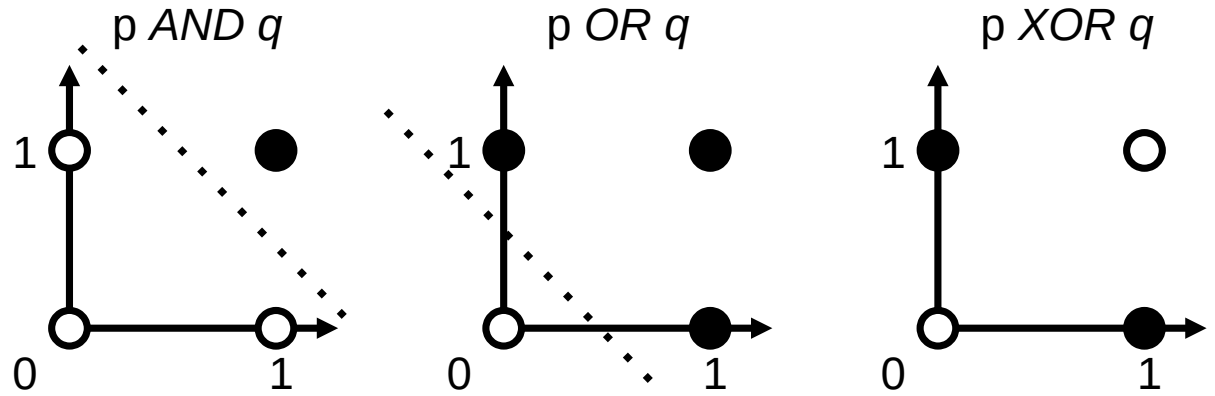
Segundo Hagan *et al.* (2014) o primeiro modelo de rede neural apresentando uma aplicação prática foi o *Perceptron* de Rosenblatt (ROSENBLATT, 1957). Além de uma ANN formada por um único neurônio usando a função de ativação *hardlim*, os autores desenvolveram uma regra de aprendizado supervisionada para a atualização dos seus parâmetros. Com isso, demonstraram a capacidade do modelo para resolver problemas de classificação e reconhecimento de padrões.

A regra de aprendizado supervisionada do *Perceptron* de Rosenblatt é dada por: Seja $\vec{\mathbf{x}}$ o vetor de entrada e t a respectiva saída esperada, a matriz de pesos atualizada $\mathbf{W^{new}}$ é dada por $\mathbf{W^{new}} = \mathbf{W^{old}} + (t-y)\vec{\mathbf{x}}$, onde $\mathbf{W^{old}}$ são os pesos (com *bias*) pré-atualização, e y é a saída obtida pelo processamento dos sinais de entrada (HAGAN *et al.*, 2014)[11]. Assim, $(t-y)$ é o erro da rede — a diferença entre saída esperada e prevista — sendo que $t = y \rightarrow \mathbf{W^{new}} = \mathbf{W^{old}}$.

Destaca-se que a capacidade de resolução de problemas do modelo é restrita. Conforme demonstração, ele é capaz de resolver um problema se ele for linearmente separável (HAYKIN, 2008; HAGAN *et al.*, 2014). Formalmente, tem-se: Demarcando as saídas da rede como pontos em um espaço euclidiano n-dimensional, deve existir pelo menos um hiperplano separando-as em dois conjuntos distintos de intersecção vazia para a rede convergir em parâmetros de solução (BISHOP, 2006; HAYKIN, 2008; HAGAN *et al.*, 2014).

Essa ideia de problemas linearmente separáveis pode ser compreendida com auxílio de um exemplo gráfico. A Figura 5 contém os mapeamentos entre entradas e saída para as funções *AND*, *OR* e *XOR* da lógica booleana. Nos planos apresentados, cada eixo é usado para corresponder às entradas dessas funções. Já o preenchimento dos pontos identifica o valor de saída, verdadeiro para preenchido e falso para transparente.

Figura 5 – Mapeamento no plano entre entrada e saída para as funções lógicas *AND*, *OR* e *XOR*.



Fonte: Elaboração própria

Percebe-se que nos casos *AND* e *OR* — linearmente separável — é possível isolar as

[11] A apresentação da regra de aprendizado do *Perceptron* de Rosenblatt tem caráter informativo. Detalhes sobre o assunto, incluindo sua interpretação geométrica, podem ser observados em Haykin (2008), Hagan *et al.* (2014) e Ekman (2021).

saídas verdadeiras das falsas por uma linha, mas para o *XOR* isso não é possível[12]. Por fim, ressalta-se que: (I) Para simplificar a apresentação foi adotada a mesma estratégia usada por Hagan *et al.* (2014) ao tratar o tema. (II) O conceito expande-se para *n*-dimensões. "Por definição, dois conjuntos de pontos x_n e y_n serão linearmente separável se existir um vetor $\vec{\mathbf{w}}$ e um escalar w_0, tal que $\vec{\mathbf{w}}^T x_n + w_0 > 0$ para todo x_n, e $\vec{\mathbf{w}}^T y_n + w_0 < 0$ para todo y_n" (BISHOP, 2006, p. 220, tradução nossa).

Dando sequência, destaca-se que o modelo em questão apenas produz saídas binárias. Essa limitação pode ser descartada ao expandi-lo para uma arquitetura *Singlelayer Feedforward* chamada de *Single Layer Perceptron* – Perceptron de Camada Única (SLP), onde cada neurônio da camada é um *Perceptron* de Rosenblatt. Entretanto, apesar de permitir a aplicação da ANN para problemas com saída *n*-dimensional, a limitação estudada anteriormente ainda é mantida (HAGAN *et al.*, 2014).

Por fim, é citado que logo após a pesquisa de Rosenblatt, Widrow e Hoff (1960) apresentaram a regra de aprendizado Delta baseada no algoritmo *Least Mean Squares* (LMS). Ela foi usada para treinar redes neurais de *Adaptative Linear Neuron* – Neurônio Linear Adaptativo (ADALINE). Esse destaque é dado, pois a rede é similar em estrutura ao modelo de Rosenblatt e sofre da mesma limitação.

2.1.1.6 Multilayer Perceptron

Redes neurais *Multilayer Perceptron* (ou *Perceptron* de Multicamadas) são modelos fundamentais na teoria do *Deep Learning* (GOODFELLOW; BENGIO; COURVILLE, 2016). Consistem de redes estruturadas na arquitetura multicamadas *Feedforward* e, em conformidade com Haykin (2008), é possível destacar três características básicas do modelo, sendo elas:

- Cada neurônio artificial que compõe a rede utiliza uma função não linear e diferenciável (que permite a aplicação da operação matemática de derivação). O uso de funções não lineares é optado, pois permite uma maior flexibilização dos resultados gerados pelos neurônios. Já a escolha de funções diferenciáveis dá-se como requisito do algoritmo utilizado para ajustar os parâmetros do modelo.
- A rede é formada por uma, ou mais camadas ocultas (como já fora visto na arquitetura *Multilayer Feedforward*);
- Apresenta alto grau de conectividade, isto é, redes MLP's costumam ser *full connected*[13].

As limitações observadas nos modelos anteriormente mencionados não estão presentes no modelo MLP. A título de curiosidade, o mapeamento da função booleana *XOR* — impossível

[12] O *Perceptron* de Rosenblatt é capaz de mapear as funções booleanas *AND* e *OR* (como foi visto na Seção 2.1.1.3), mas não a função *XOR*, uma vez que ela não é linearmente separável.

[13] É aquela em que os elementos de uma camada x possuem ligações com todos os elementos da camada $x+1$. Em contrapartida, redes que não possuem essa característica são denominadas *partial connected* (HAGAN *et al.*, 2014).

usando uma rede SLP, ou ADALINE — pode ser resolvido com uma rede MLP formada por uma camada oculta de dois neurônios e um neurônio de saída (HAGAN *et al.*, 2014; GOODFELLOW; BENGIO; COURVILLE, 2016; EKMAN, 2021).

Outro ponto de destaque dessa rede neural é o seu algoritmo de aprendizado. Segundo Hagan *et al.* (2014), Rosenblatt e Widrow sabendo da limitação de seus modelos, tentaram propor redes multicamadas. Todavia, nenhum foi capaz de generalizar a regra de aprendizado delas. Para esse propósito, o algoritmo chamado *backpropagation* só seria explorado em meados de 1980, após ser redescoberto independentemente por diferentes pesquisadores[14] e popularizado pelo livro *Parallel Distributed Processing* (RUMELHART; MCCLELLAND, 1986).

Sendo assim, "[...] o *Multilayer Perceptron* treinado pelo algoritmo de *backpropagation*, é atualmente a rede neural mais utilizada" (HAGAN *et al.*, 2014, paginação irregular, tradução nossa) e por isso, apresenta-se uma discussão sobre esse algoritmo. Do ponto de vista geral, ele é uma regra de aprendizado supervisionado baseada em métodos de aprendizado por correção de erro amplamente fundamentados sob as teorias da otimização matemática de funções.

Levando em consideração a rede neural $\hat{g}$ como um modelo matemático para aproximar a função f, para um conjunto de dados de treino

$$S = \left\{ (\vec{\mathbf{x}^1}, \vec{\mathbf{y}^1}), (\vec{\mathbf{x}^2}, \vec{\mathbf{y}^2}), ..., (\vec{\mathbf{x}^m}, \vec{\mathbf{y}^m}) \right\} \tag{19}$$

define-se $\vec{\mathbf{x}^i}$ como uma entrada n-dimensional arbitrária e $\vec{\mathbf{y}^i}$ a sua respectiva saída em S. Assim, é destacado que a saída ótima de f e a saída prevista por $\hat{g}$ são dadas por

$$f(\vec{\mathbf{x}^i}) = \vec{\mathbf{y}^i} \tag{20}$$

$$\hat{g}(\vec{\mathbf{x}^i}) = \vec{\hat{\mathbf{y}}^i} \tag{21}$$

Ao utilizar ambos os valores de saída é possível calcular o erro do modelo $\hat{g}$.

O erro consiste de uma mensuração da disparidade entre a aproximação do modelo e os resultados ótimos (HAGAN *et al.*, 2014). Intuitivamente quanto menor for o erro, melhor são as aproximações dos valores em relação às saídas esperadas. Todavia, para essa quantificação é necessário definir uma função de erro, comumente chamada de função de perda em DL (em inglês, respectivamente, *error function* e *loss function*).

Existem diferentes funções que podem ser usadas para esse propósito (diferença simples entre valores, quadrado da diferença entre valores, módulo, etc.) e suas aplicações não são limitadas a área de redes neurais, sendo amplamente estudadas em outros domínios. Enquanto, para o algoritmo *backpropagation* destaca-se o uso do *Mean Square Error* – Erro Quadrático Médio (MSE), pois atende certas propriedades — como ser diferenciável (HAGAN *et al.*, 2014) — e foi utilizado em trabalhos iniciais da área.

[14] Segundo Hagan *et al.* (2014, paginação irregular, tradução nossa), aparentemente a primeira descrição de um algoritmo para treinar redes multicamadas estava contida na tese de Paul Werbos em 1974. Os pesquisadores independentes citados foram David Rumelhart, Geoffrey Hinton e Ronald Williams, David Parker e Yann Le Cun.

Portanto, para todo o conjunto de dados S o cálculo do MSE é feito por

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}(y_j^i - \hat{y}_j^i)^2 \tag{22}$$

Referente a notação, o índice subscrito j é usado para identificar os componentes de um vetor n-dimensional e o índice sobrescrito i identifica o vetor no conjunto S. Por fim, destaca-se que o erro pode ser aproximado para uma única entrada $\vec{\mathbf{x}^{\mathbf{k}}}$ via

$$E[k] = \sum_{j=1}^{n}(y_j^k - \hat{y}_j^k)^2 \tag{23}$$

onde k não é uma variável, mas um identificador para referenciar a entrada.

O MSE depende dos valores de $\vec{\mathbf{y}^{\mathbf{i}}}$ e $\vec{\hat{\mathbf{y}}^{\mathbf{i}}}$ obtidos pelas funções f e $\hat{g}$. Sendo f o objetivo de aproximação, a única função que pode variar os seus valores gerados é $\hat{g}$, alterando os parâmetros do modelo. Portanto, a ideia que fundamenta o algoritmo *backpropagation* consiste na otimização dos parâmetros de cada neurônio do modelo $\hat{g}$ visando minimizar o erro entre as saídas esperada e obtida.

Minimizar o erro tem o mesmo significado que encontrar um ponto mínimo na função de erro. Esse estudo pertence a área da otimização matemática (e numérica) e pode ser resolvido através da aplicação de um algoritmo conhecido como descida do gradiente (*gradient descent*)[15]. Portanto, em sequência são definidos e destacados brevemente alguns conceitos necessários para compreender esse algoritmo.

Primeiramente, são definidas as funções genéricas

$$h : \mathbb{R}^n \to \mathbb{R}, h(\vec{\mathbf{x}}) \tag{24}$$

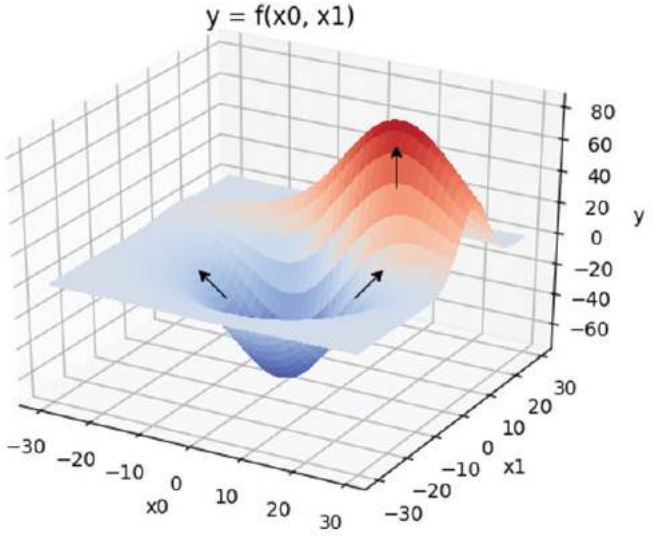
$$\nabla h : \mathbb{R}^n \to \mathbb{R}^n, \nabla h(\vec{\mathbf{x}}) = \begin{pmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} & \dots \frac{\partial h}{\partial x_n} \end{pmatrix} \tag{25}$$

onde $h(x_1, x_2, \dots, x_n) = h(\vec{\mathbf{x}})$. A função (25) é chamada de gradiente da função (24), sendo geometricamente interpretada por: Seja $\vec{\mathbf{x}} = (x_1, x_2, \dots, x_n)$ pertencente ao domínio de h, ∇h é o vetor no mesmo espaço cuja direção indica a variação de maior incremento em h (BISHOP, 2006; HAGAN *et al.*, 2014). Por fim, é mencionado que se $\nabla h = 0$ então a região da função em questão possivelmente é um ponto mínimo.

Uma exemplificação é feita com auxílio do gráfico de uma função arbitrária $f : \mathbb{R}^2 \to \mathbb{R}$ apresentado pela Figura 6, onde as setas ilustram a direção e inclinação do maior incremento em três pontos. Sendo assim, destaca-se que: (I) Os gradientes calculados nos pontos são as projeções vetoriais das setas no plano; (II) Na direção oposta às setas é possível encontrar um ponto mínimo — região em que $(x_0, x_1) = (0, 0)$.

[15] O algoritmo não é perfeito. É impossível garantir que seja encontrada a solução ótima para todas as funções possíveis. Uma discussão somente desse tópico necessita de tempo e conhecimento prévio em otimização matemática. Sendo assim, o tema é comentado superficialmente para embasar o necessário ao escopo do trabalho.

Figura 6 – Gráfico indicando vetor de maior incremento para três pontos de uma função em $\mathbb{R}^3$.



Fonte: Ekman (2021, paginação irregular)

Com essa informação, a partir de um vetor de entrada $\vec{\mathbf{x}}_{\mathbf{k}}$ qualquer em h é desejado deslocar-se em direção ao mínimo da função. Calculando $\vec{\mathbf{x}}_{\mathbf{k+1}} = \vec{\mathbf{x}}_{\mathbf{k}} - \alpha \nabla h(\vec{\mathbf{x}}_{\mathbf{k}})$ é possível obter um vetor $\vec{\mathbf{x}}_{\mathbf{k+1}}$ indicando um ponto (possivelmente) mais próximo do mínimo da função (HAYKIN, 2008; HAGAN *et al.*, 2014). Repetido o procedimento n vezes é possível "caminhar" pelo domínio da função e aproximar-se de um valor mínimo — esse é o algoritmo da descida do gradiente.

Para os modelos MLP's a função originalmente minimizada é o MSE aproximado em $E[k]$, e o gradiente é calculado para ser usado em relação a cada parâmetro do modelo. O objetivo então é encontrar, usando os dados do conjunto S, iterativamente pela descida do gradiente os valores dos parâmetros que reduzem a função de erro, isto é, aproximam do seu valor de mínimo. Esse procedimento iterativo é também descrito como descida do gradiente estocástica[16].

Formalmente a atualização dos pesos e *bias* que minimizam a função de erro é dada por

$$\hat{w}_{i,j}^{m} = w_{i,j}^{m} - \alpha \frac{\partial E}{\partial w_{i,j}^{m}} \tag{26}$$

onde o peso atualizado $\hat{w}_{i,j}^{m}$ — referente ao parâmetro j, do neurônio i, na camada m — é obtido subtraindo do peso original $w_{i,j}^{m}$ o respectivo elemento do vetor de gradiente do erro aproximado

$$\frac{\partial E}{\partial w_{i,j}^{m}} \tag{27}$$

multiplicado por uma taxa aprendizado α[17].

O algoritmo de aprendizagem pode ainda ser visualizado através da sua divisão em duas etapas de processamento. A primeira consiste em calcular os valores de saída da rede neural —

[16] Estocástico, pois o modelo é otimizado usando um erro aproximado iterativamente para cada entrada. Existem métodos não estocásticos, como o *backpropagation em batch* explicitado em Hagan *et al.* (2014).

[17] Configuração definida pelo projetista da rede. Impacta na capacidade da rede em resolver o problema esperado — detalhes em (HAGAN *et al.*, 2014).

fase conhecida como *forward pass*. Já na segunda, os valores das saídas são usados para calcular as derivadas parciais necessárias, iniciando da última camada até a camada de entrada — fase conhecida como *backward pass* (HAYKIN, 2008; HAGAN *et al.*, 2014).

O *backward pass* é realizado no sentido contrário do *forward pass* por restrição da regra de aprendizado. Expandindo a equação[18] (25), observa-se que para calcular as derivadas parciais dos pesos de um neurônio situado na camada m são necessárias as informações das derivadas parciais de todos os neurônios da camada $m+1$ (exceto para os neurônios de saída). Finalizando, cita-se que generalizações do algoritmo são feitas por Haykin (2008) e Hagan *et al.* (2014), já uma apresentação simplificada está em Ekman (2021).

2.1.2 Aspectos de Processamento de Imagens

O processamento de imagens consiste na aplicação de operações para manipulação e transformação de imagens. Como exemplo de operações são citadas a correção de exposição, o equilíbrio de cores, a redução de ruído, o aumento da nitidez, o redimensionamento e a detecção de bordas. É uma importante área de estudo, uma vez que diversas aplicações em visão computacional a utilizam no pré-processamento de dados para obtenção de resultados aceitáveis (SZELISKI, 2010).

No que diz respeito ao presente trabalho, faz-se necessário compreender alguns tópicos fundamentais dessa área, uma vez que a pesquisa refere-se a um problema de classificação de imagens. Sendo assim, em sequência são apresentados conceitos referentes à representação de imagens — úteis para a correta manipulação desse tipo de dado — e o modelo de ruído utilizado para degradação de imagens aplicado pelos experimentos.

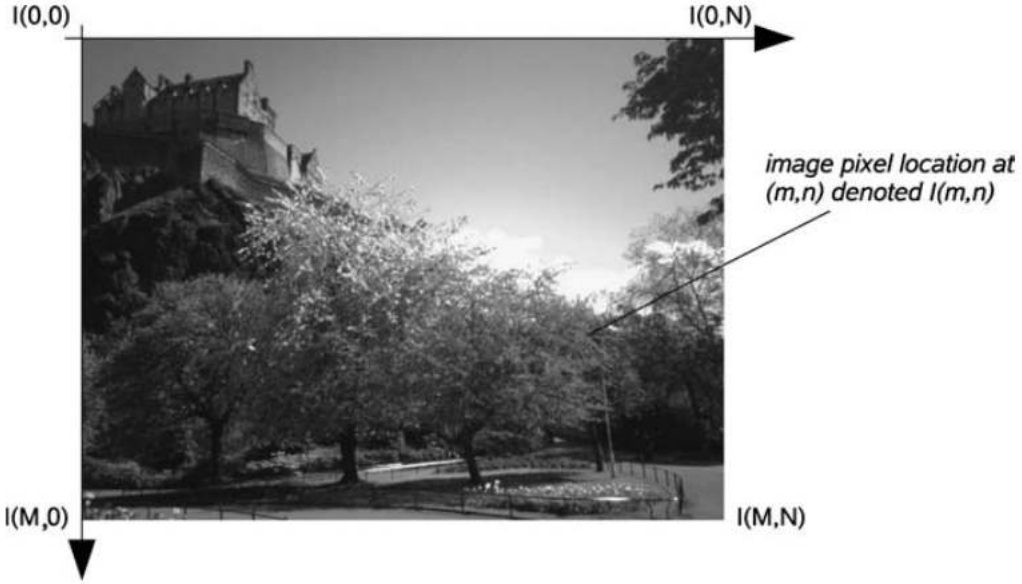
2.1.2.1 Representação de Imagens Digitais

As imagens digitais são utilizadas para armazenar e representar dados com semântica visual. Elas são usualmente obtidas através do uso de sensores — como as câmeras —, ou podem ser criadas sinteticamente por procedimento matemáticos. Além disso, segundo Solomon e Breckon (2010, p. 1, tradução nossa) “uma imagem digital pode ser considerada como uma representação discreta de dados que possui tanto informações espacial (*layout*), quanto de intensidade (cor).”.

Ainda em conformidade com os autores supramencionados, o *layout* de uma imagem digital I é formado por dados discretizados a partir da amostragem de um sinal contínuo. Esses dados podem ser associados a posições em um sistema de coordenadas cartesiano, vide Figura 7. Sendo assim, cada posição (m,n) em $I(m,n)$ corresponde a um *pixel* — dado base que forma a imagem.

[18] Por motivos de simplificação, não será apresentada a expansão da regra de aprendizado, todavia ela foi aplicada na implementação dos modelos usados pela pesquisa experimental.

Figura 7 – Espaço de coordenadas para uma imagem digital de dimensões *M*x*N*.



Fonte: Solomon e Breckon (2010, p. 2)

Sobre a intensidade, uma imagem pode conter um ou mais canais que definem a cor de seus *pixels*. Sendo assim, durante o processo de exibição de imagens os valores de seus *pixels* são associados às cores conforme uma função de mapeamento. Considerado isso, destaca-se a existência de diferentes mapeamentos para a intensidade, sendo os mais comuns listados em sequência e exemplificados visualmente pela Figura 8.

Figura 8 – Apresentação de uma mesma imagem em três mapeamentos de cores.

Fonte: Elaboração própria.

- Imagem binária: cada *pixel* pode assumir o valor 0, ou 1. O mapeamento de cores é feito atribuindo a coloração preta para *pixels* de valor 0 e a coloração branca para o restante;
- Imagem em escala de cinza: cada *pixel* possui um valor discreto que pertencente ao intervalo $[0, 255]$. O mapeamento para a cor é feita pela atribuição de tonalidades de cinza proporcionais ao *pixel*, sendo preto associado ao 0 e branco ao 255;

- Imagem RGB: cada *pixel* corresponde a um vetor tridimensional onde cada componente representa uma tonalização, seguindo o mesmo padrão de intervalo do mapeamento anterior, para uma das seguintes cores: vermelho (*Red*), verde (*Green*), ou azul (*Blue*).

2.1.2.2 Modelo de Ruído Estacionário Aditivo Gaussiano

Um modelo de ruído consiste de um mecanismo matemático para adição de perturbações em uma imagem. Através de um processo conhecido como degradação, os *pixels* da imagem original $I(m,n)$ são alterados conforme valores de ruído gerados por uma função matemática. Sendo assim, após aplicação do modelo sob a imagem, obtém-se como resultado final uma imagem $G(m,n) \neq I(m,n)$ (SOLOMON; BRECKON, 2010; FORSYTH; PONCE, 2011; CATTIN, 2016).

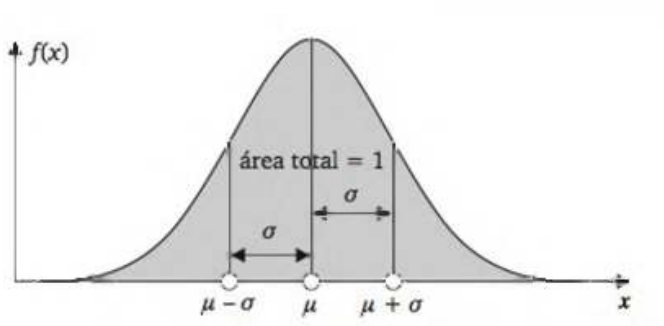
Como exemplo de modelo comum para degradação de imagens cita-se o Modelo de Ruído Estacionário Aditivo Gaussiano. Sua compreensão pode ser feita através da explicitação de suas propriedades, sendo ela apresentada em sequência. Finalizando, a Figura 9 apresenta uma ilustração de uma mesma imagem em seu estado original e após a degradação para exemplificar o resultado obtido da aplicação do modelo de ruído estudado.

Figura 9 – Efeito visual da degradação de uma imagem.

Fonte: Elaboração própria.

O modelo em questão é dito ser: (I) Estacionário, pois os ruídos gerados não variam com o decorrer do tempo — a função é independente do parâmetro tempo; (II) Aditivo, pois no processo de degradação é somado um valor de ruído ao valor original de cada *pixel* da imagem; (III) Gaussiano, uma vez que os valores de ruído para uma mesma imagem são amostrados de uma (única) distribuição de probabilidade normal (Figura 10), frequentemente com média zero.

Figura 10 – Representação da distribuição de probabilidade normal com seus parâmetros μ (média) e σ (desvio padrão).



Fonte: Barbetta, Reis e Bornia (2010, p. 153)

Sobre a distribuição de probabilidade normal, os eixos x e y identificam respectivamente um valor de análise e sua probabilidade de ser observado[19]. Por fim, destaca-se que como parâmetros do modelo de ruído é permitida a escolha do desvio padrão σ (ou std) (FORSYTH; PONCE, 2011). Dessa forma, quanto maior for essa medida, maiores são as chances de obter valores distantes da média e consequentemente uma maior degradação da imagem.

2.2 PROJETO E RESULTADO DOS EXPERIMENTOS

Dá-se sequência ao trabalho apresentando a pesquisa experimental realizada. Para cada experimentação citada na Seção 1.2, a seção apresenta a sua descrição seguida pela análise de seus resultados. Por fim, é importante ressaltar que a execução dessa pesquisa segue uma passagem linear, isto é, a realização de um experimento depende do experimento anterior.

2.2.1 Descrição do Experimento I

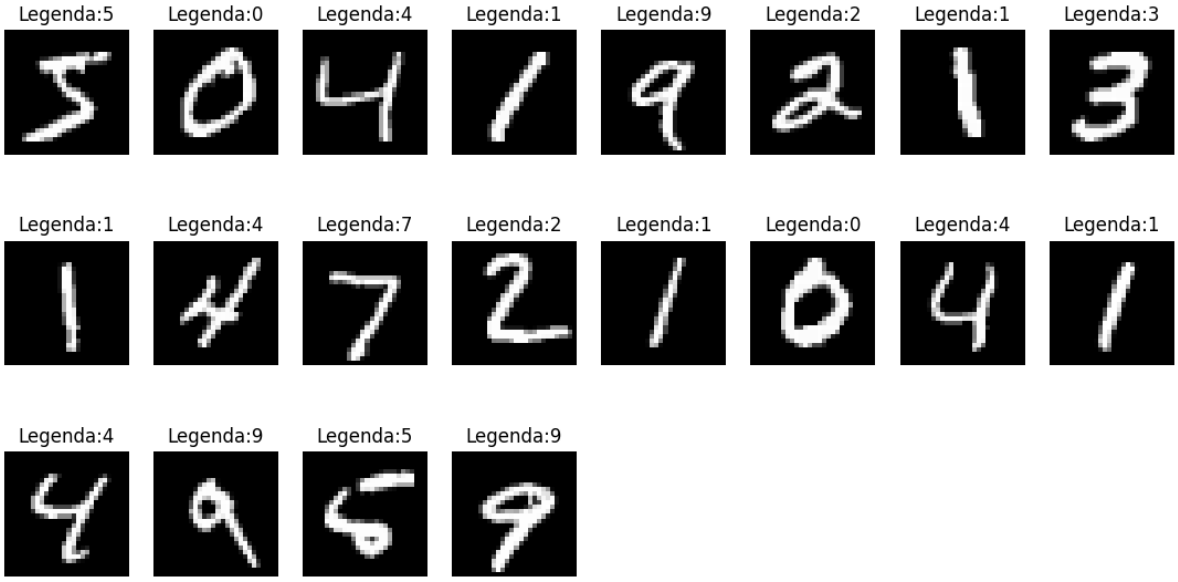
O primeiro experimento consiste em avaliar o desempenho de um modelo de rede neural MLP desenvolvido para solucionar o problema de classificação de dígitos manuscritos desconsiderando imagens ruidosas. As imagens são obtidas a partir do conjunto de dados MNIST. Sobre esse conjunto, destaca-se que a fonte dos dígitos é uma mistura da escrita proveniente de funcionários do *American Census Bureau* e de estudantes do ensino médio americano.

O conjunto de dados é formado por 70 mil imagens — 60 mil formando o conjunto de treino e 10 mil o conjunto de validação — em escala de cinza de dígitos manuscritos e suas respectivas legendas de classificação. Cada imagem possui dimensões 28 por 28 e representa apenas um único dígito. Já a legenda consiste de um número inteiro no intervalo de 0 à 9 identificando respectivamente o dígito apresentado na imagem. Exemplificando, na Figura 11 são observadas imagens pertencentes ao MNIST, bem como suas respectivas legendas/rótulos.

Por fim, é mencionado que o conjunto apresenta uma distribuição aproximadamente homogênea (não há uma disparidade significativa na quantidade de dados por categoria de

[19] Para calcular a probabilidade de obter valores de $x \in [a,b]$ basta calcular a área da função $f(x)$ nesse limitada nesse intervalo (BARBETTA; REIS; BORNIA, 2010).

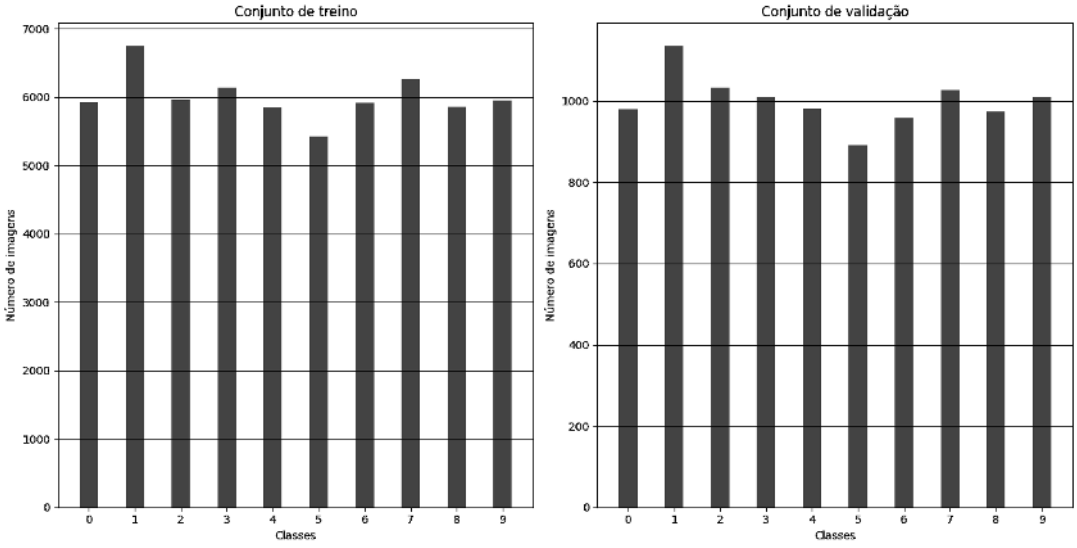
Figura 11 – Exemplos de imagens pertencentes ao conjunto de dados MNIST e suas respectivas legendas.



Fonte: Elaboração própria. Imagens retiradas do conjunto MNIST.

dígito), conforme Figura 12. Sendo assim, não é observado o problema de desbalanceamento de classes (citado na Seção 2.1.1.4), facilitando o processo de generalização do modelo para as diferentes classes que constituem o problema de classificação em questão.

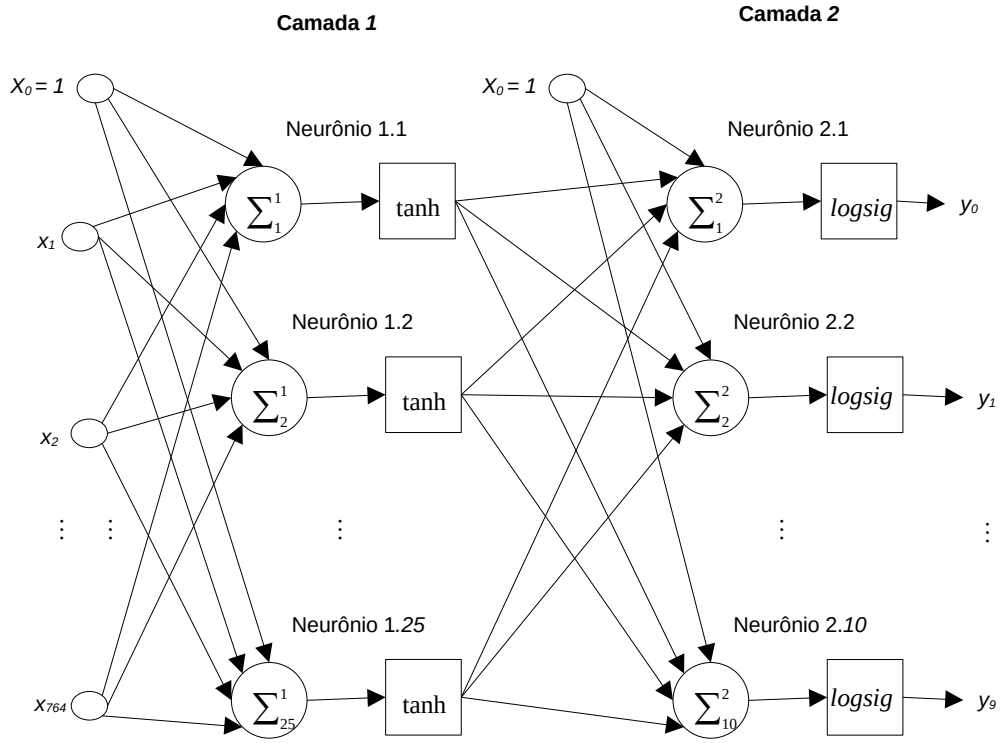
Figura 12 – Distribuição dos dados do conjunto MNIST (de treino e de validação) por classe.



Fonte: Elaboração própria.

Considerado isso, dá-se início a apresentação dos detalhes do modelo de classificação utilizado no experimento. A rede neural desenvolvida é baseada no modelo de mesmo propósito apresentado em Nielsen (2015) e Ekman (2021). Consiste de um modelo MLP *full connected* composto por duas camadas de neurônios, sendo a primeira referente ao processamento da entrada da rede e a última utilizada para o processamento das saídas da classificação do modelo (Figura 13).

Figura 13 – Arquitetura do modelo de classificação desenvolvido, baseada em Ekman (2021).



Fonte: Elaboração própria.

No que diz respeito à camada de entrada, ela recebe um vetor $\vec{\mathbf{x}}$ composto por 785 sinais de entrada. O primeiro sinal é associado ao valor fixo $x = 0$ usado para cálculo dos *biases* dos neurônios da camada, e o restante são associados aos dados da imagem passada como entrada ao modelo — correspondem aos *pixels*. Para mapear uma imagem para um vetor é possível concatenar os *pixels* obtidos em uma leitura linha à linha[20].

Destaca-se que os vetores de imagens do conjunto de dados são pré-processados para serem usados pelo modelo. Nessa operação os dados são normalizados, sendo cada valor referente a um *pixel* dividido por 255. Apesar de não ser um procedimento essencial para o desenvolvimento do modelo, dessa forma são obtidos vetores formados por componentes de ponto flutuante com magnitude controlada, o que auxilia na obtenção de melhores resultados (EKMAN, 2021).

Dando sequência, a primeira camada (oculta) de processamento é formada por 25 neurônios. A escolha dessa quantidade é feita pelo seu uso nos trabalhos anteriormente mencionados — sendo o valor destacado como uma escolha arbitrária de projeto. Considerado isso, destaca-se que a função de ativação utilizada é a tangente hiperbólica (*tanh*), definida como

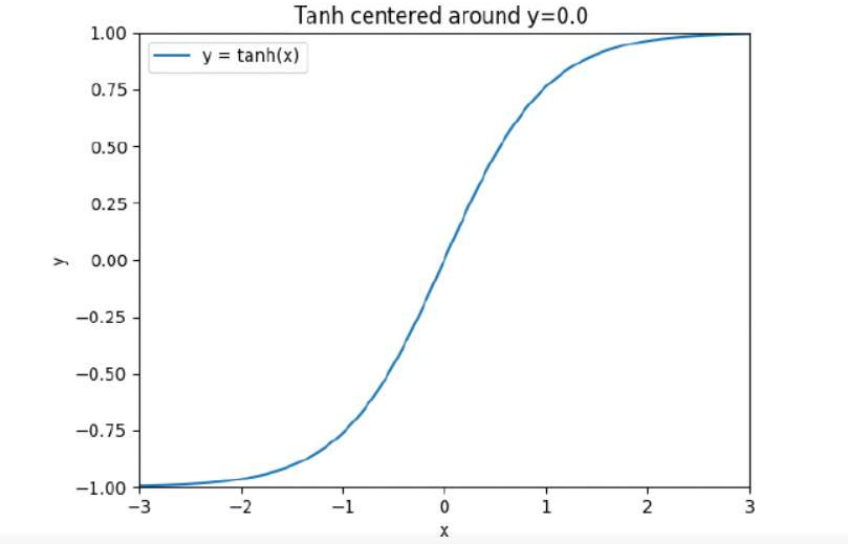
$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{28}$$

Um recorte do gráfico da função é exibido pela Figura 14. A sua escolha é dada, pois assemelha-se a função *hardlim*, porém é diferenciável, possibilitando seu uso no algoritmo *backpropaga-*

[20] Outros mapeamentos podem ser usados, como a leitura de coluna à coluna. O importante durante o desenvolvimento e aplicação do classificador é adotar um padrão único.

tion.

Figura 14 – Recorte da função tangente hiperbólica centrada em $y = 0$



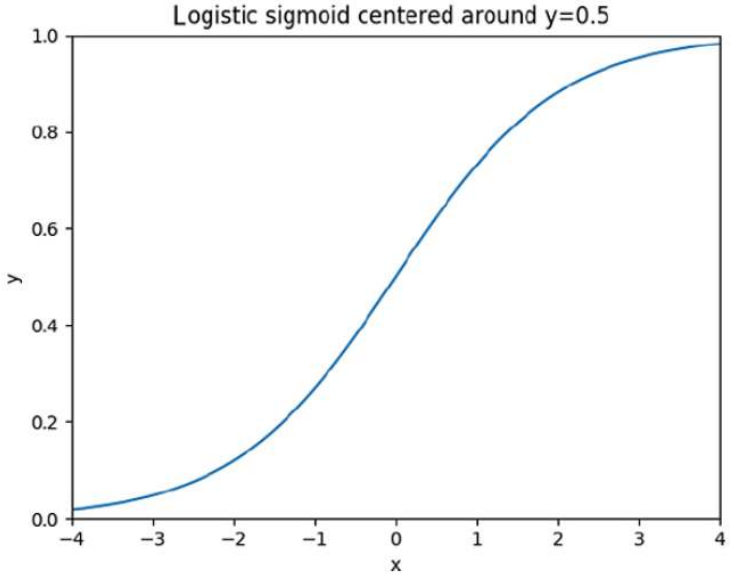
Fonte: Ekman (2021, paginação irregular)

No que diz respeito à segunda (e última) camada de processamento, ela é composta por 10 neurônios que utilizam a função de ativação sigmóide logística (*logsig*), definida por

$$logsig(x) = \frac{1}{1+e^{-x}} \tag{29}$$

O recorte do gráfico da função é exibido pela Figura 15. Essa função é escolhida, pois pode ser interpretada como uma distribuição de probabilidade (EKMAN, 2021). Assim, cada saída é interpretada como a probabilidade da entrada ser classificada como uma categoria (dígito) específica.

Figura 15 – Recorte da função sigmóide logística centrada em $y = 0.5$



Fonte: Ekman (2021, paginação irregular)

Explicitando, seja

$$\vec{\mathbf{y}} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_9 \end{pmatrix} \tag{30}$$

o vetor de saída obtido pelo processamento de uma entrada pelo classificador de imagens. Dado o componente y_i, para o modelo em questão o seu valor refere-se à probabilidade da imagem passada à rede neural ser classificada como o dígito i. Logo, o resultado da classificação corresponde ao índice do componente de maior valor.

Apresentada a arquitetura do modelo de classificação, faz-se necessário destacar o seu processo de aprendizado. O treino é feito usando o algoritmo *backpropagation* estocástico e a função de erro MSE aproximada para cada entrada. Seguindo a configuração adotada por Ekman (2021) como escolha de projeto, a taxa de aprendizado utilizada é definida como $\alpha = 0,01$. Por fim, destaca-se outra configuração necessária para o treino de modelos de rede neural — a época (do inglês *epoch*).

Esse valor identifica a quantidade de iterações realizadas sobre todo o conjunto de treino. É importante, pois quanto mais épocas passam, mais atualizações são realizadas e (possivelmente) um melhor resultado é obtido (HAYKIN, 2008; HAGAN *et al.*, 2014; GOODFELLOW; BENGIO; COURVILLE, 2016). Portanto, para o treino do modelo final do experimento são definidas a passagem de 10 épocas — consideradas suficientes para produzir resultados satisfatórios para a pesquisa em questão.

Por fim, para a etapa de validação do modelo a acurácia da classificação é usada como métrica desempenho —. Essa mensuração é definida em termos do percentual da quantidade de acertos em relação à quantidade total de dados usados como entrada. Por exemplo, sabendo que a quantidade total de dados de validação usada é 10 mil, dada uma taxa de classificações corretamente previstas c, a acurácia de validação é matematicamente dada por $acc = c/10000,00$.

2.2.2 Análise dos Resultados do Experimento I

Os resultados do experimento I visam destacar principalmente o desempenho do classificador de imagens — mensurado pela acurácia — sobre o conjunto de dados de validação provenientes do MNIST. Considerado isso, na Tabela 1 são apresentadas as acurácias de treino e validação do modelo em relação a cada passagem de época considerada durante o processo de aprendizado da rede neural em questão.

Apesar do interesse da pesquisa estar relacionado com o desempenho do modelo no conjunto de validação após a finalização do processo de aprendizado, como justificativas para apresentação da acurácia em relação ao treino e às épocas intermediárias são mencionadas respectivamente: (I) permite confirmar a generalização do problema pelo modelo; (II) possibilita observar a evolução do desempenho do classificador de imagens conforme treinamento.

Tabela 1 – Acurácia (treino e validação) de classificação em relação às épocas de treino.

Época	Acurácia média — treino (%)	Acurácia média — validação (%)
1	85,63	91,57
2	92,03	92,40
3	92,74	92,46
4	93,20	92,95
5	93,56	92,97
6	93,84	93,32
7	94,06	93,50
8	94,20	93,46
9	94,40	93,67
10	94,57	93,61

Fonte: Elaboração própria.

Com os dados apresentados, primeiramente é possível destacar a capacidade do modelo em aprender e resolver o problema proposto. Em apenas uma época de treino o desempenho em ambos os cenários considerados podem ser julgados como satisfatórios e melhores que uma adivinhação aleatória (que possui probabilidade próxima de 10% de ser correta). Dentre as justificativas para isso estão a simplicidade do problema em relação às outras tarefas de classificação com mais categorias e dados diversos.

Além disso, com exceção das épocas 8 e 10 para o conjunto de validação, a passagem de épocas implica no aumento da acurácia — o que faz sentido, uma vez que o modelo realiza mais adaptações aos seus parâmetros. Todavia, é possível perceber que após a 2ª época o ganho em desempenho torna-se menor, nunca superando 0,75%. Isso pode decorrer do fato do modelo já ser capaz de produzir bons resultados em 1 época de treino, reduzindo a intensidade do processo de otimização realizado durante o algoritmo *backpropagation*.

Retomando as exceções supramencionadas, a quebra no padrão de incremento de acurácia pode ser entendida pela adaptação de pesos ser realizada com base nos dados de treino. Sendo assim, os parâmetros obtidos podem impactar reduzindo ligeiramente a acurácia de validação na passagem de uma época. Entretanto, se o modelo está generalizando os dados, essa diferença não será significativa e o passar de mais épocas a suprime como observado nos resultados.

Também é interessante notar que ambas as taxas de desempenho que representam um sucesso considerável na classificação das imagens. Sendo assim, o modelo está sendo capaz de generalizar o processo de classificar dígitos manuscritos e não está apenas decorando os dados de entrada para a produção das saídas esperadas. Inclusive, ao final das 10 épocas os desempenhos observados no treino e na validação são próximos.

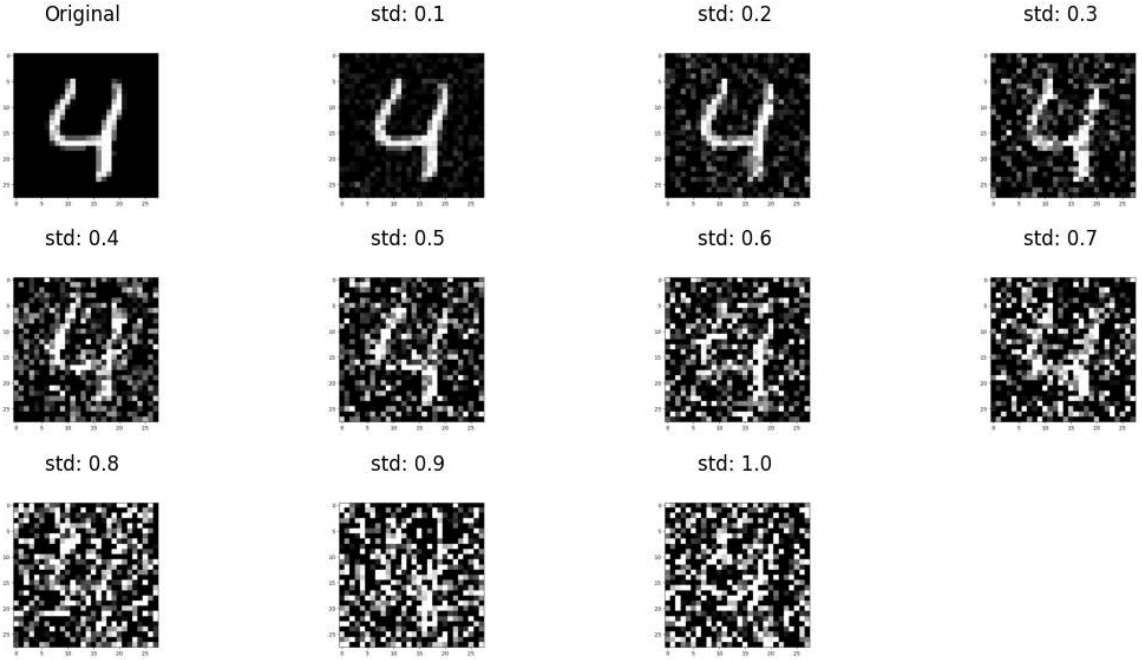
2.2.3 Descrição do Experimento II

O segundo experimento é resumido em analisar o impacto no desempenho da classificação gerado pela adição de ruído às imagens utilizadas como entrada para a validação do modelo. Sendo assim, dado o classificador treinado na Seção 2.2.1, diferentes versões do conjunto de validação são criadas para a analise. Cada versão consiste na introdução de perturbações em diferentes níveis criadas usando o modelo de ruído descrito na Seção 2.1.2.2.

A variação em níveis de ruído é feita pela variação do desvio padrão (std), parâmetro usado pelo mecanismo que gera as perturbações. Sendo assim, para o experimento são definidos dez níveis de desvio padrão, sendo eles de 0,1 a 1,0, variando 0,1 por nível. Dessa forma, quanto maior o desvio padrão, maior é a degradação gerada na imagem, dificultando o reconhecimento da informação representada pelo dado.

Portanto, para cada nível de ruído σ é realizado o seguinte procedimento em todas as imagens de validação: seja o vetor $\vec{\mathbf{x}}$ os dados da imagem normalizada, um vetor $\vec{\mathbf{u}}$ de mesma dimensionalidade é gerado, onde cada componente é um valor aleatoriamente obtido com o modelo gerador de ruídos configurado com desvio padrão σ. O vetor com ruído é computado pela soma $\vec{\mathbf{x}} + \vec{\mathbf{u}}$, sendo aplicada a limitação dos seus componentes para pontos flutuantes no intervalo $[0, 1]$ visando manter o padrão de entradas aceito pelo classificador de imagens. Com isso, a Figura 16 exemplifica as imagens obtidas[21] pela realização desse procedimento para cada nível de ruído usado na pesquisa.

Figura 16 – Imagem (dígito 4) do conjunto de validação para diferentes níveis de ruído.



Fonte: Elaboração própria.

2.2.4 Análise dos Resultados do experimento II

Os resultados do experimento abordam o impacto gerado pela introdução de ruído nas imagens utilizadas como entrada do classificador desenvolvido. Considerados os diferentes níveis de ruído destacados na Seção 2.2.3, a Tabela 2 apresenta a média da acurácia obtida em 10 etapas de validação para cada conjunto de dados ruidosos. O cálculo da média é feito, pois os valores de ruído obtidos são aleatoriamente distribuídos e produzem conjuntos diferentes para cada validação.

[21] Para obter a o resultado visual, deve-se converter o vetor de dados para uma imagem.

Tabela 2 – Acurácia média de classificação para conjuntos de validação com diferentes níveis de ruído.

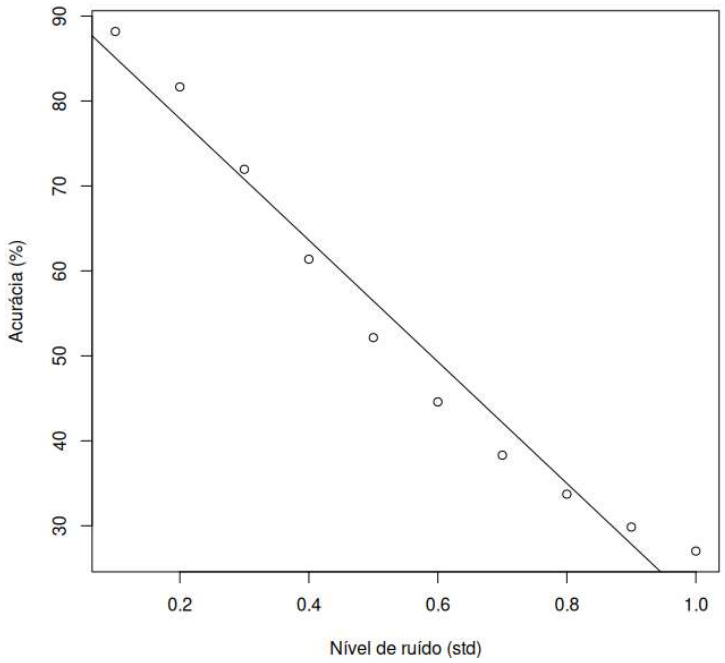
Nível de ruído (std)	Acurácia média de classificação (%)
0,1	88,18
0,2	81,66
0,3	71,96
0,4	61,38
0,5	52,15
0,6	44,58
0,7	38,31
0,8	33,72
0,9	29,85
1,0	27,02

Fonte: Elaboração própria.

Com base nos dados obtidos, ao comparar as mensurações da Tabela 2 com o desempenho de validação final do experimento anterior (93,61%), é perceptível o decremento na taxa de classificações corretas. Além disso, destaca-se que quanto maior o incremento no nível de ruído, maior é o decremento da acurácia em relação ao valor base — o que é esperado, pois torná-se mais difícil para o classificador utilizar o conhecimento generalizado a partir de imagens em que os dígitos são nítidos para prever entradas degradadas.

Aproveitando a menção da variação da acurácia em relação aos níveis de ruídos, é interessante notar que o decremento não segue uma relação perfeitamente linear. A Figura 17 apresenta um gráfico relacionando as duas medidas, bem como a reta de regressão linear simples[22] para os pontos. Com isso, é possível perceber que apesar da relação não-linear, aproximar um modelo de reta explicita o decremento da acurácia conforme incremento de ruído nos dados.

Figura 17 – Gráfico de dispersão da acurácia em relação aos níveis de ruído, em conjunto da aproximação de modelo de regressão linear.



Fonte: Elaboração própria.

[22] Dado um conjunto de pontos de uma função desconhecida, a regressão aproxima a relação entre as variáveis dependentes e independentes por uma função linear (BARBETTA; REIS; BORNIA, 2010).

Sendo assim, conforme constatado em trabalhos citados na Seção 1.4, a introdução de ruído aos dados de entrada de um modelo de classificação que não é robusto a essas perturbações impacta negativamente no seu desempenho. No experimento em questão, para o pior caso (maior nível de ruído) o desempenho chega a cair mais que 65%. Dessa forma, a taxa de sucesso é consideravelmente reduzida e a utilização do modelo de classificação é prejudicada.

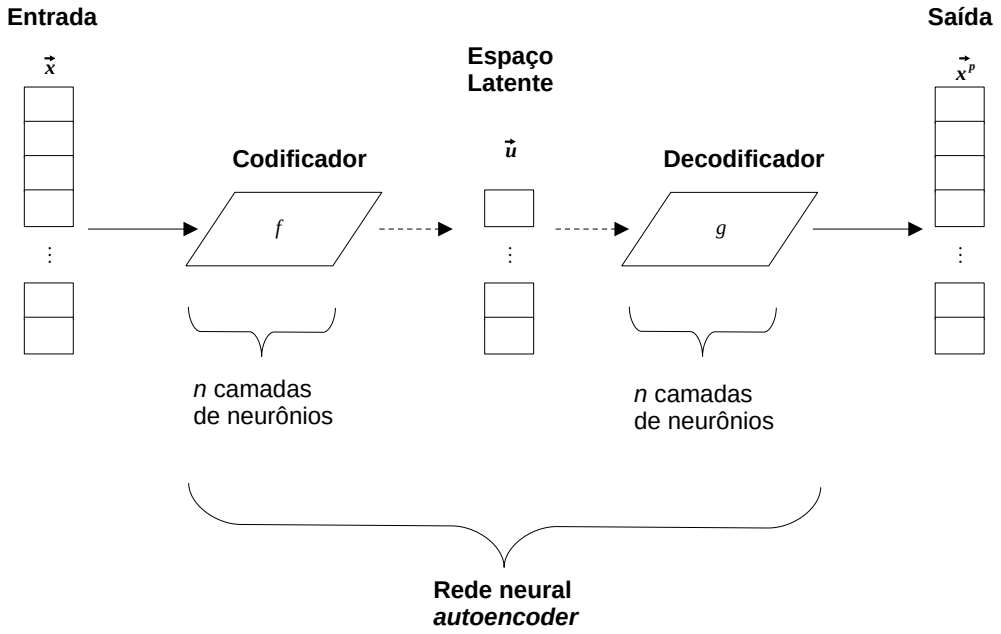
2.2.5 Experimento III

O terceiro experimento baseia-se na avaliação do desempenho da classificação de imagens ruidosas quando elas são pré-processadas por uma rede neural para remoção de ruído. Para cada uma das versões dos conjuntos de validação ruidosos obtidas na Seção 2.2.3, um removedor de ruído (de mesma configuração inicial) é treinado especificamente para detectar e remover aquele nível de perturbação das imagens.

O modelo para essa tarefa consiste na rede MLP estruturada como um *autoencoder*. Esse tipo de ANN pode ser entendido pela sua divisão em três partes: um codificador, o espaço latente e um decodificador. A ideia original por trás desse modelo é, dada uma entrada $\vec{\mathbf{x}}$, aproximar as seguintes funções (em uma única rede neural): $f : \mathbb{C}^n \to \mathbb{C}^m, f(\vec{\mathbf{x}}) = \vec{\mathbf{u}}$ e $g : \mathbb{C}^n \to \mathbb{C}^m, g(\vec{\mathbf{u}}) = \vec{\mathbf{x}}$, com $m < n$. (HAYKIN, 2008; GOODFELLOW; BENGIO; COURVILLE, 2016).

A função f é chamada de codificador e sua funcionalidade é produzir uma saída $\vec{\mathbf{u}}$ que representa as características significativas de $\vec{\mathbf{x}}$, porém sendo um vetor com menor dimensionalidade. O vetor $\vec{\mathbf{u}}$ recebe a nomenclatura de espaço latente. Já a função g é chamada de decodificador e seu objetivo é reconstruir uma saída próxima $\vec{\mathbf{x^p}}$ do dado original a partir do espaço latente (Figura 18). (GOODFELLOW; BENGIO; COURVILLE, 2016).

Figura 18 – Exemplificação da arquitetura *autoencoder*.



Fonte: Elaboração própria.

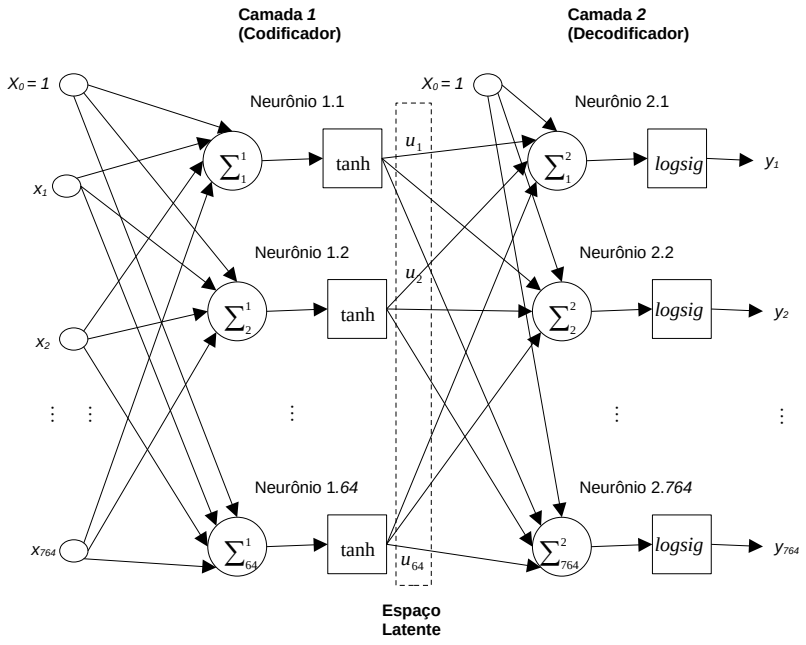
Em termos de arquitetura, a rede *autoencoder* recebe uma entrada $\vec{\mathbf{u}}$ que é processada por n camadas de neurônios, objetivando a redução de dimensionalidade e obtenção do espaço latente $\vec{\mathbf{u}}$. Para isso, é necessário reduzir os neurônios conforme a introdução de camadas, forçando uma camada oculta geradora de $\vec{\mathbf{u}}$. Após isso, deseja-se reconstruir uma aproximação do dado original. Para isso, são adicionadas camadas que incrementam a quantidade de neurônios até que a última seja equivalente a primeira — para a saída e a entrada terem a mesma dimensionalidade.

Sobre o treino, os modelos *autoencoders* recaem no aprendizado não supervisionado. Idealmente, apresenta-se a entrada original e espera-se que o modelo aprenda a gerar uma saída similar a ela. Assim, é possível usar o algoritmo *backpropagation* para que o modelo reduza uma função de erro expressando a diferença entre os componentes do vetor de entrada em relação aos do vetor de saída (GOODFELLOW; BENGIO; COURVILLE, 2016; EKMAN, 2021).

Considerado isso, o uso dessa arquitetura para a remoção de ruído requer uma adaptação. Ao invés de apresentar o dado original como entrada ao modelo, apresenta-se uma versão da entrada com perturbações (EKMAN, 2021). Dessa forma, o modelo é otimizado para reduzir a diferença entre um vetor ruidoso e o mesmo vetor sem ruído, implicando no aprendizado de um modelo para remoção de ruído.

Os modelos *autoencoders* usados na pesquisa são baseados em uma única arquitetura, exemplificada na Figura 19. Adaptada de Ekman (2021), a rede é composta por duas camadas de neurônios e recebe 785 sinais de entrada (pelos mesmos motivos do modelo da Seção 2.2.1) que são encaminhados à 64 neurônios, configuração arbitrariamente definida. Logo, o espaço latente é o vetor composto pelos potenciais de ativação desses 64 neurônios. Por fim, a última camada possui 784 neurônios para que a saída tenha a mesma dimensão da imagem de entrada.

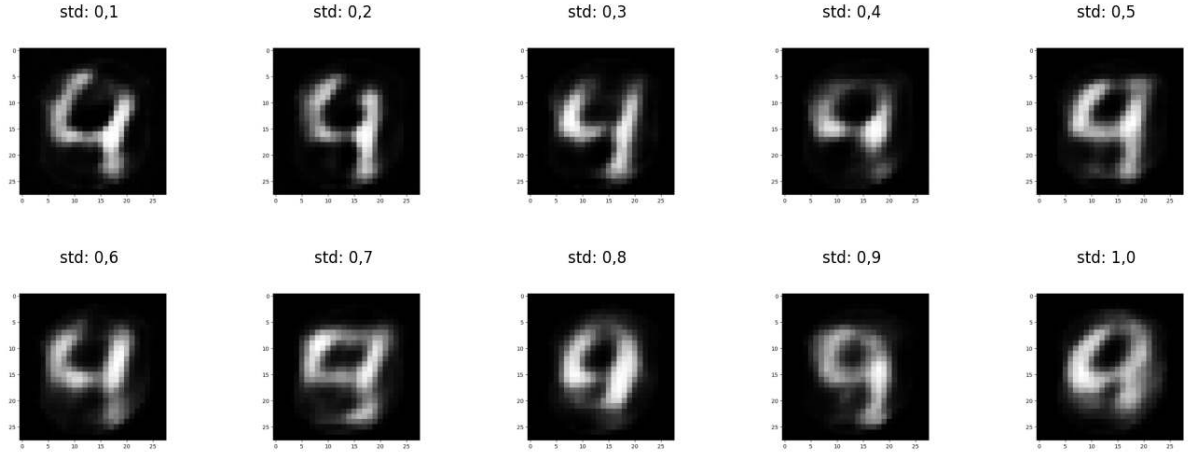
Figura 19 – Arquitetura *autoencoder* MLP usada implementada pelos modelos de remoção de ruído.



Fonte: Elaboração própria.

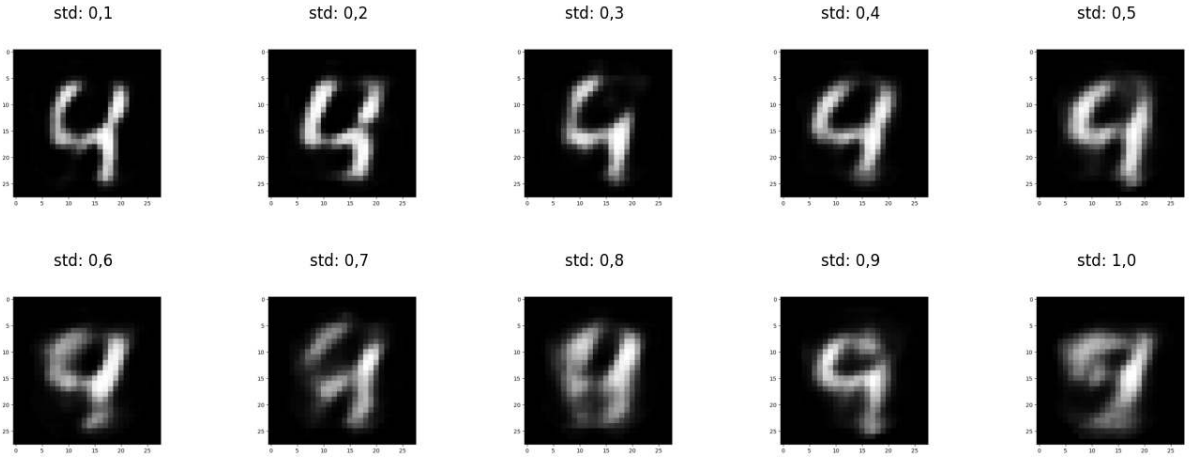
As demais configurações — funções de ativação, taxa de aprendizado, algoritmo de aprendizagem, conjunto de dados — seguem as definições do modelo de classificação. Porém, para as épocas são geradas versões dos removedores de ruído (de cada nível de ruído) treinados em 1, 5 e 10 épocas — possibilitando avaliar o impacto no desempenho da classificação conforme aprimoramento do pré-processamento obtido pelas redes *autoencoders*. A título de exemplificação, nas Figuras 20, 21 e 22 são destacadas imagens ruidosas após processamento de seus removedores de ruído.

Figura 20 – Saída dos removedores de ruído (1 época) para entrada ruidosa referente ao dígito 4.



Fonte: Elaboração própria.

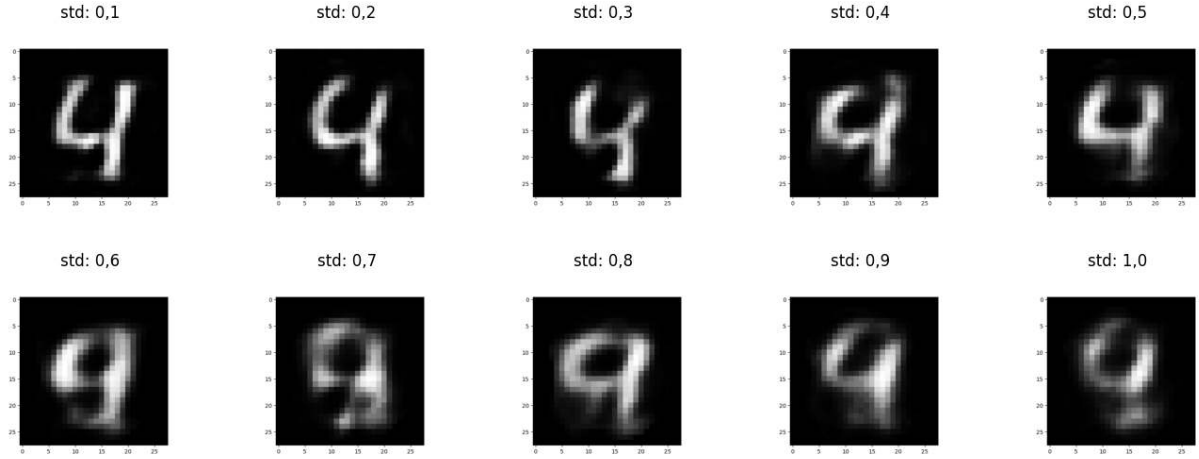
Figura 21 – Saída dos removedores de ruído (5 épocas) para entrada ruidosa referente ao dígito 4.



Fonte: Elaboração prórpia.

Por fim, destaca-se que a fase de validação e verificação da acurácia da remoção de ruído não é realizada, uma vez que não existe uma saída correta a ser contabilizada pela métrica. Poder-se-ia realizar uma validação levando em consideração a análise da medida usada para treinar o modelo, todavia o objetivo da pesquisa está focado na acurácia da classificação de imagens.

Figura 22 – Saída dos removedores de ruído (10 épocas) para entrada ruidosa referente ao dígito 4.



Fonte: Elaboração prórpia.

2.2.6 Análise dos Resultados do Experimento III

Como resultados são apresentadas as acurácias médias obtidas ao usar os modelos de remoção de ruído para pré-processar as imagens ruidosas passadas ao modelo de classificação. Similarmente aos resultados do experimento II, a mensuração é feita em relação a média obtida em 10 etapas de validação para cada conjunto de dados ruidosos. Todavia, são levados em consideração modelos *autoencoders* treinados em 1, 5 e 10 épocas. Considerado isso, os resultados são destacados em sequência pela Tabela 3.

Tabela 3 – Acurácia média de classificação para conjuntos de validação (com diferentes níveis de ruído) pré-processados pelos respectivos modelos de remoção de ruído treinados em diferentes épocas.

Nível de ruído (std)	**Acurácia média de classificação (%) - removedor de ruído 1 época**	**Acurácia média de classificação (%) - removedor de ruído 5 épocas**	**Acurácia média de classificação (%) - removedor de ruído 10 épocas**
0,1	76,92	81,56	86,73
0,2	73,71	82,51	84,76
0,3	73,35	76,49	82,38
0,4	61,69	73,09	77,47
0,5	56,57	62,58	68,24
0,6	55,89	56,94	63,71
0,7	48,24	58,75	58,08
0,8	52,85	56,53	54,58
0,9	44,37	53,11	53,17
1,0	39,70	43,75	50,85

Fonte: Elaboração própria.

Primeiramente, com base nos resultados obtidos é possível destacar que na maioria dos casos a passagem de mais épocas de treino implica na obtenção de melhores taxas de classificação. Intuitivamente, quanto melhor treinado, melhores são as imagens processadas pelo modelo de remoção de ruído e melhores são as taxas de classificação. Dessa forma, usar e aprimorar as redes neurais em questão como uma etapa de pré-processamento das imagens a serem classificadas demonstra ser benéfico, principalmente para os casos de maior presença de ruído.

Porém, não são todos os cenários em que o pré-processamento incrementa a acurácia de classificação. Comparando os resultados com os dados da Tabela 2 (Seção 2.2.3) nota-se que para o menor nível de ruído o desempenho obtido sem a remoção de ruído é maior do que com a sua aplicação. Aplicar o pré-processamento nesse caso gerou um decremento aproximado de 11,26%, 6,63% e 1,45% nas acurácias respectivamente associadas às épocas 1, 5 e 10.

Outra informação interessante a ser citada é o fato de que aplicar os removedores de ruído para diferentes níveis de perturbação não indica um valor constante de incremento na acurácia (em relação à Tabela 2). Por exemplo, para os modelos treinados em 1 época o ganho na classificação foi próximo de 19% para $std = 0,8$, cerca de 4% para $std = 0,5$ e em torno de 12% onde $std = 1,0$. Além disso, não necessariamente quanto menor a intensidade de ruído, melhor é a generalização do modelo de remoção de ruído implicando em incremento da acurácia de classificação.

Considerado isso, aproveita-se para mencionar os piores e melhores cenários de aplicação do pré-processamento para a classificação. Para ambas as épocas o pior cenário é identificado na aplicação do removedor de ruído para $std = 0,1$. Já os melhores cenários para a passagem das diferentes épocas são destacados:

- 1 época - $std = 0,8$: Observa-se um incremento de 19,14% na acurácia de classificação;
- 5 épocas - $std = 0,9$: Destaca-se um incremento de 23,27% na acurácia de classificação;
- 10 épocas - $std = 1,0$: Cita-se um incremento de 23,84% na acurácia de classificação.

Por fim, são analisados os incrementos (e decrementos) obtidos pela aplicação do modelo de remoção de ruído para o pré-processamento de dados sob uma perspectiva geral. Dessa forma, é destacado que em média são obtidos aprimoramentos na acurácia de classificação de 5,45%, 11,65% e 15,12% respectivamente usando as redes treinadas com 1, 5 e 10 épocas. Portanto, o uso de modelos relativamente simples (em estrutura e treino) apresenta melhorias para o problema de classificação de dígitos manuscritos provenientes de imagens ruidosas.

2.2.7 Experimento IV

O quarto e último experimento estuda a possibilidade de utilizar o processamento do classificador de imagens para auxiliar o processo de aprendizagem dos modelos de remoção de ruído. A inspiração para essa abordagem é feita com base em trabalhos do campo de pesquisa *Multitask Learning* – Aprendizado Multitarefas (MTL). Segundo Caruana (1997, p. 41, tradução nossa):

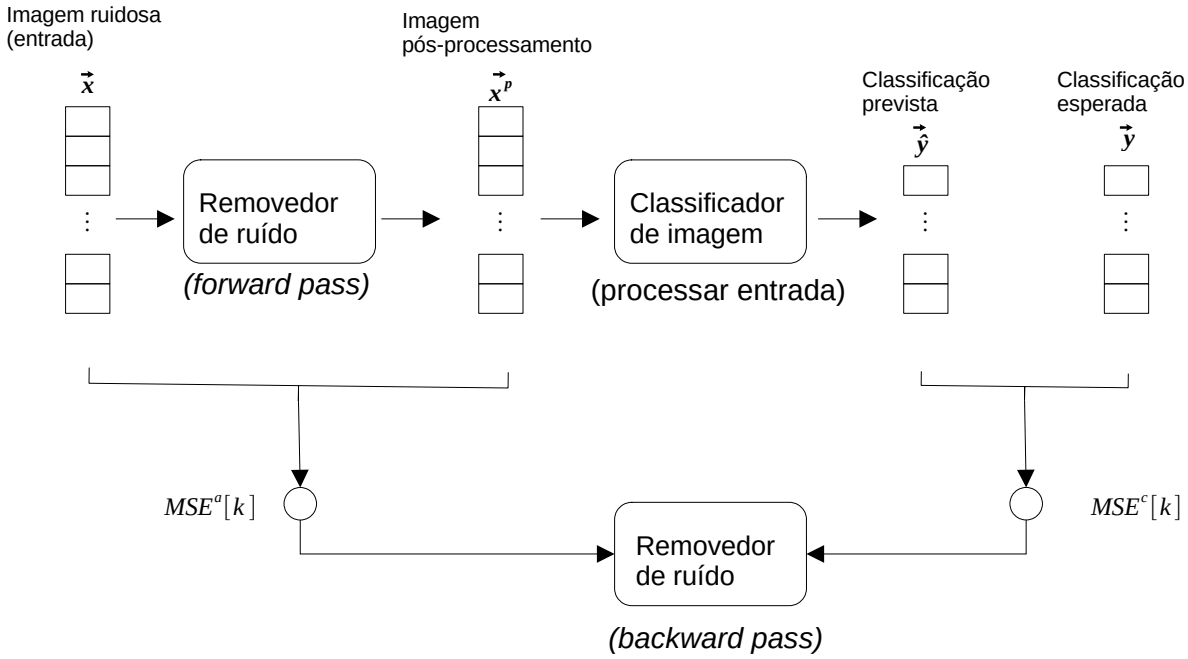
> *Multitask Learning* (MTL) é um mecanismo de transferência indutiva cujo objetivo principal é melhorar o desempenho da generalização. MTL melhora a generalização aproveitando informações específicas de domínio contidas nos sinais de treinamento de tarefas relacionadas. Ele faz isso treinando (modelos) para tarefas em paralelo usando uma representação compartilhada.

Isto é, existe a possibilidade de treinar modelos cujas tarefas possuam relação de forma conjunta visando melhores resultados.

Todavia, destaca-se novamente que a abordagem estudada na presente pesquisa não envolve o treino de duas redes neurais conjuntamente. O classificador de imagens já é conhecido e previamente treinado. O foco está em utilizar o resultado de seu processamento durante o treino de modelos para remoção de ruído de forma que sejam geradas imagens pré-processadas que induzam melhores taxas de classificação.

Sendo assim, o experimento explora a utilização do valor do erro aproximado $E^c[k]$ do classificador de imagens junto do erro aproximado $E^a[k]$ do removedor de ruído. Dessa forma, para cada imagem $\vec{\mathbf{x}}$ de um conjunto de dados de treino com ruído, durante o aprendizado de um removedor de ruído a saída processada por ele é passada ao classificador, antes de realizar o passo *backward* do algoritmo *backpropagation*, e o valor de $E^c[k]$ obtido é usado no seu treino (Figura 23).

Figura 23 – Esquemático da adaptação explorada para treinar um modelo de remoção de ruído.



Fonte: Elaboração prórpia.

O valor obtido é utilizado para excitar os neurônios da última camada de processamento do removedor de ruído. O intuito é introduzir um valor de erro mais significativo e representativo para a tarefa de classificação no modelo de remoção de ruído. Dessa forma, idealmente é buscada a possibilidade da rede *autoencoder* aprender a priorizar o processamento de saídas que impacta na redução do erro e incremento da acurácia do classificador de imagens.

Considerado isso, se explorada a Equação (26) apresentada na Seção 2.1.1.6 é possível destacar que para os modelos desenvolvidos na pesquisa — usando o erro MSE aproximado ($E[k]$) e a função de ativação *logsig* —, para cada neurônio i da última camada é calculado um

termo de erro (EKMAN, 2021) definido por

$$eN_i = (-y_i + \hat{y}_i).logsig'(v_i) \tag{31}$$

A equação consiste na derivação do erro aproximado para o removedor de ruído em relação ao neurônio i da última camada vezes dois[23]. O valor obtido é então utilizado na equação Equação (26) para realizar a passagem *backward* do algoritmo *backpropagation*.

Sendo assim, a alteração explorada para introduzir a medida de erro da classificação $E^c[k]$ é definida pela equação

$$\hat{eN}_i = (\phi(-y_i + \hat{y}_i) + \omega E^c[k]).logsig'(v_i) \tag{32}$$

onde ϕ e ω são duas configurações que devem ser definidas durante o projeto da rede neural. A ideia é realizar uma soma ponderada com as medidas do classificador de imagens e do removedor de ruído.

Todavia, destaca-se que a ponderação é realizada em relação a um dos produtos de eN_i e não com a equação inteira. Essa escolha é dada, pois

$$-y_i + \hat{y}_i \tag{33}$$

é a derivada da função de erro MSE aproximado (multiplicado por dois) considerando um neurônio de saída i. Observada a relação da derivação com a taxa de crescimento de uma função, é buscado potencializar a variação da diferença entre a saída esperada y_i e a saída obtida $\hat{y}_i$ através de uma soma ponderada.

Finalizando, destaca-se que a ponderação de informações relacionadas às medidas de erros dos modelos é inspirada (principalmente) nos trabalhos de Liao *et al.* (2016) e Teichmann *et al.* (2018). Ambos abordam problemas distintos do escopo da pesquisa, mas aplicam métodos de MTL. São trabalhos em que diferentes problemas são solucionados por redes neurais que compartilham uma única métrica de erro obtida pela soma de seus erros individuais.

Com isso, o experimento consiste em utilizar a adaptação do algoritmo *backpropagation* variando as configurações ϕ e ω para replicar procedimento similar a Seção 2.2.5. Todavia, ao invés de treinar os modelos de remoção de ruído com diferentes épocas, o processo de aprendizagem é dado com uma época — limitação feita, uma vez que para cada par de ϕ e ω são treinados 10 *autoencoders* (um para cada nível de ruído).

A título de especificação, o conjunto de valores usados para ϕ e ω é respectivamente dado pelos pares de valores

$$\begin{pmatrix} (+1.0;-1.0),(+0,8;-0,2),(+0,5;-0,5), \\ (+1.0;-0,5),(+1.0;+1.0),(+0,5;0,5), \\ (+1.0;+0,5),(+0,8;+0,2),(+1.0;\pm 0,5) \end{pmatrix} \tag{34}$$

[23] A multiplicação é feita para eliminar uma operação de divisão e simplificar o cálculo aplicado durante o treino. O mesmo é realizado em Ekman (2021).

escolhidos arbitrariamente via experimentação, observando como as mudanças impactam no desempenho do classificador de imagens. Além disso, destaca-se que o último par de valores (onde $\omega = \pm 0,5$) é traduzido como

$$(-y_i + \hat{y}_i) \geq 0 \rightarrow \omega = 0,5; (-y_i + \hat{y}_i) < 0 \rightarrow \omega = -0,5 \tag{35}$$

Sendo assim, a configuração representa um ω flexível calculado para cada imagem usada durante o algoritmo de treino.

2.2.8 Análise dos Resultados do Experimento IV

Como resultados do último experimento são apresentadas as acurácias de classificação de imagens pré-processadas por modelos de remoção de ruído treinados usando a adaptação do algoritmo *backpropagation*, conforme especificado do experimento IV. Para cada configuração de valores (ϕ, ω) são apresentadas as médias de 10 validações da taxa de classificações corretas para cada nível de ruído. Por fim, relembra-se que os modelos *autoencoders* do experimento atual são treinados em apenas 1 época. Visto isso, os resultados em questão são apresentados pela Tabela 4.

Tabela 4 – Acurácia média de classificação para conjuntos de validação (com diferentes níveis de ruído) pré-processados pelos respectivos modelos de remoção de ruído treinados com configurações (ϕ, ω).

Configuração (ϕ, ω)	**Acurácia média de classificação (%) por nível de ruído (std)**									
	0,1	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1,0**
$(\phi = +1,0, \omega = -1,0)$	75,80	73,96	72,72	68,29	68,22	66,31	59,35	52,64	49,42	41,46
$(\phi = +0,8, \omega = -0,2)$	80,38	79,62	72,01	65,84	68,09	61,36	55,32	52,40	45,95	42,91
$(\phi = +0,5, \omega = -0,5)$	70,05	74,12	71,20	69,86	69,85	68,78	62,67	54,77	49,79	42,14
$(\phi = +1,0, \omega = -0,5)$	82,66	80,82	80,29	75,75	70,28	67,75	62,88	60,69	52,65	48,26
$(\phi = +1,0, \omega = +1,0)$	66,17	63,33	52,09	26,90	42,85	15,07	15,03	28,58	19,56	15,91
$(\phi = +0,5, \omega = +0,5)$	57,30	60,60	34,15	42,93	15,64	9,74	9,72	24,91	9,72	9,74
$(\phi = +1,0, \omega = +0,5)$	69,82	70,55	55,74	51,29	47,94	33,60	30,57	29,64	31,84	28,24
$(\phi = +0,8, \omega = +0,2)$	72,49	72,57	56,75	54,69	54,26	46,16	38,73	39,24	34,13	34,13
$(\phi = +1,0, \omega = \pm 0,5)$	73,64	74,81	60,12	60,66	57,10	51,35	50,44	42,90	43,33	37,73

Fonte: Elaboração própria.

Primeiramente, é possível destacar que a mudança dos valores utilizados produz resultados de classificação distintos. Isso é esperado, uma vez que as configurações ϕ e ω são aplicadas diretamente no algoritmo de aprendizado dos modelos removedores de ruído. É interessante perceber que a escolha pode tanto produzir boas taxas de classificação — onde $\phi = 1$ e $\omega = -0,5$ —, quanto decrementar consideravelmente a acurácia obtida — onde $\phi = 0,5$ e $\omega = 0,5$.

Comparando os resultados obtidos com os dados da aplicação de removedores de ruído de 1 época treinados sem o algoritmo adaptado, apresentados na Tabela 3, é possível destacar:

- O par $(\phi = 1, \omega = -0,5)$ apresentou maiores incrementos em relação ao modelo padrão. No pior caso, foi ganho cerca de 5,7% em acurácia — nível de ruído 0,1. Já nos melhores casos, a acurácia foi incrementada em mais de 14% — níveis de ruído 0,4 e 0,7. Considerado isso, em média é obtido um incremento de 9,89% nas acurácias de classificação com os removedores de ruído adaptados;

- O par $(\phi = 0,5, \omega = 0,5)$ apresentou os maiores decrementos em relação ao modelo padrão. No pior caso, foi reduzido aproximadamente 46% em acurácia — nível de ruído 0,6. Já no melhor caso, a acurácia é decrementada em 13% — nível de ruído 0,2. Dessa forma, em média é obtido um decremento superior a 30% nas acurácias de classificação ao utilizar removedores de ruído treinados com esses valores de ϕ e ω.

Dando sequência, destaca-se que dentre as configurações apresentadas, as quatro primeiras repercutem majoritariamente em incrementos nas acurácias de classificação em relação ao uso de modelos *autoencoders* treinados pelo *backpropagation* padrão. Já as demais configurações impactam negativamente no desempenho observado. Interessante notar que os resultados positivos são observados quando $\omega < 0$ e o inverso também é observado.

Outro ponto a ser discutido está relacionado ao uso de $\omega = \pm 0,5$ — uma configuração variável adaptada ao longo da execução do algoritmo de treino. Conforme os dados apresentados na Tabela 4 e comparados com a Tabela 3, essa configuração não impactou positivamente o desempenho do classificador de imagens. Sendo assim, os indícios iniciais apontam para a preferência ao uso de valores fixos para as configurações ω.

Por fim, é feita uma breve comparação entre o melhor desempenho identificado na Tabela 4 $(\phi = 1, \omega = -0,5)$ com os desempenhos obtidos ao aplicar os removedores de ruído padrões treinados em 5 e 10 épocas (destacados pela Tabela 3). Em comparação ao uso dos modelos de 5 épocas, exceto para o nível de ruído 0,8, ao utilizar os modelos treinados com a adaptação do algoritmo *backpropagation* foram obtidas acurácias de classificação superiores do que com os modelos padrões treinados em 5 épocas.

Esse resultado é destacado, uma vez que mesmo com uma menor passagem de épocas os modelos adaptados apresentaram maiores benefícios em termos de classificações corretas. Isso é interessante, pois em termos computacionais são realizadas menos operações para treinar 1 época usando o algoritmo adaptado do que para treinar em 5 épocas com o algoritmo padrão. Dessa forma, o uso do método explorado na pesquisa apresenta indícios de possíveis melhoras na eficiência do processo de aprendizado.

Já ao comparar o mesmo cenário, porém para o modelo padrão treinado em 10 épocas, são observadas situações diferentes. O modelo de 10 épocas implica em acurácias superiores às obtidas com o modelo adaptado de 1 época para seis níveis de ruídos distintos — 0,1 à 0,4, 0,9 e 1,0. Já nos casos restantes o uso do modelo adaptado implicou em métricas melhores. No melhor cenário — nível de ruído 0,8 —, observa-se uma diferença positiva de 7% na acurácia obtida com o modelo adaptado em relação ao modelo padrão de 10 épocas.

Sendo assim, é destacada a possível relevância prática em explorar e estudar o uso do algoritmo de aprendizado adaptado para obter melhores métricas no problema de classificação em questão. Com os resultados obtidos no experimento observa-se que existe um impacto a ser estudado pelo uso da abordagem explorada na pesquisa. Considerado isso, é finalizada a discussão dos resultados e é dado encaminhamento às conclusões finais do trabalho.

3 CONCLUSÃO

Com o presente trabalho foi possível observar a exploração de um problema de classificação de imagens de dígitos manuscritos com técnicas de *Deep Learning*. Além da avaliação da rede neural para essa tarefa, foram propostos cenários adversos — através da introdução de ruído nos dados — para avaliar o desempenho do modelo. Por fim, foi estudada a aplicação de pré-processamento dos dados — via rede neural *autoencoder* — objetivando destacar melhorias no desempenho do modelo de classificação.

Além disso, baseado no estudo do algoritmo *backpropagation* e tomando como inspiração estudos na área de *Multitask Learning*, uma adaptação ao processo de aprendizado das redes neurais de remoção de ruído é apresentada e inicialmente explorada. Dessa forma, foi levantada a possibilidade de introduzir uma medida de erro proveniente da classificação de imagens no treino desses modelos visando o incremento da acurácia de classificação.

Portanto, dentre os resultados obtidos é possível destacar principalmente:

- A introdução de ruído nas imagens de entrada de um modelo de classificação que não é robusto à essas perturbações implica na redução da sua capacidade de inferir saídas corretamente;
- Ao utilizar modelos para remoção de ruído durante etapa de pré-processamento percebe-se o incremento na taxa de sucesso da rede neural de classificação, principalmente para dados com perturbações intensas;
- Observada a correlação entre as tarefas de remoção de ruído e classificação, através de um processo exploratório observa-se que o uso de informações provenientes do modelo de classificação para auxiliar o processo de aprendizado de modelos de remoção de ruído incrementa o sucesso de classificação.

O último resultado mencionado é apresentado como diferencial da presente pesquisa. Os dados obtidos empiricamente demonstram a existência de uma caminho a ser estudado que pode aprimorar a eficiência de modelos de redes neurais relacionadas. Além da obtenção de melhores taxas de classificação, a adaptação do algoritmo *backpropagation* explorada indica a possibilidade de reduzir as épocas necessárias para obtenção de resultados finais similares.

Dessa forma, é possível destacar a realização dos objetivos geral e específicos estipulados pelo trabalho. Considerando isso, é importante relembrar que a pesquisa quanto aos seus objetivos consistiu principalmente na exploração de uma ideia. Logo, os dados e resultados obtidos podem ser utilizados para a elaboração de pesquisas explicativas que visam formalizar e compreender os detalhes da adaptação do processo de aprendizagem de redes neurais para problemas relacionados.

Sendo assim, como primeira frente de desenvolvimento para pesquisas futuras destaca-se a reprodução e ampliação dos experimentos conduzidos no trabalho visando testificar e formalizar os indícios que foram observados. Exemplificando, poder-se-ia estudar os mesmos

cenários, porém fazendo o uso de modelos treinados — principalmente os removedores de ruído adaptados — em mais épocas. Além disso, uma análise formal do algoritmo explorado é de extrema importância, pois visa analiticamente compreender suas capacidades e explicar os resultados obtidos na prática.

Outra possibilidade de trabalho futuro está no estudo de cenários similares ao apresentado na pesquisa. Isto é, ao invés de explorar o problema de classificação de imagens de dígitos manuscritos, outros problemas plausíveis de aplicação de técnicas de DL em que são usadas mais de uma rede neural podem ser estudados. Assim, é possível ampliar a aplicação do algoritmo de treino adaptado para outros problemas na tentativa de verificar se resultados similares são observados.

Por fim, para concluir a pesquisa é explicitado o seu potencial de aplicação e evolução dos resultados obtidos. Ao verificar o sucesso inicial da regra de aprendizado explorada, diferentes possibilidades de pesquisas sobre o tema são criadas. Além disso, com o desenvolvimento e concretização dos indícios apresentados os conceitos explorados podem ser usados para desenvolver soluções de DL eficientes voltadas para aplicações reais.

REFERÊNCIAS

BALDOMINOS, Alejandro; SAEZ, Yago; ISASI, Pedro. A survey of handwritten character recognition with MNIST and EMNIST. **Applied sciences**, v. 9, n. 15, p. 16, ago. 2019.

BARBETTA, Pedro Alberto; REIS, Marcelo Menezes; BORNIA, Antonio Cezar. **Estatística para cursos de engenharia e informática**. 3. ed. São Paulo: Atlas, 2010. 416 p.

BISHOP, Christopher M. **Pattern Recognition and Machine Learning**. 1. ed. New York, NY: Springer, 2006. 738 p.

BOTTOU, L. *et al.* **Comparison of classifier methods**. *In*: **IAPR**. [*S.l.*]: IEEE, 1994. v. 3, p. 77–82.

BUADES, Antoni.; COLL, Bartomeu.; MOREL, Jean-Michel. A review of image denoising algorithms, with a new one. **Multiscale modeling and simulation**, v. 4, n. 2, p. 490–530, 2005.

CARUANA, Rich. Multitask Learning. **Machine learning**, v. 28, n. 1, p. 41–75, 1997.

CATTIN, Philippe. **Image Restoration**. 2016. Disponível em: https://docplayer.net/85966756-Image-restoration-introduction-to-signal-and-image-processing-prof-dr-philippe-cattin-miac-university-of-basel-april-19th-26th-2016.html. Acesso em: 13 nov. 2023.

DODGE, Samuel; KARAM, Lina**. Understanding how image quality affects deep neural networks**. *In*: **2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)**. [*S.l.*]: IEEE, 2016. p. 1–6.

EKMAN, Magnus. **Learning deep learning**. 1. ed. Boston, MA: Addison Wesley, 2021. 752 p.

FORSYTH, David A.; PONCE, Jean. **Computer Vision**. 2. ed. Upper Saddle River, NJ: Pearson, 2011. 761 p.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. London, England: MIT Press, 2016. 775 p.

GU, Shuhang; TIMOFTE, Radu**. A brief review of image denoising algorithms and beyond**. *In*: **Inpainting and Denoising Challenges**. [*S.l.*]: Springer International Publishing, 2019. p. 1–21.

HAGAN, Martin T.; DEMUTH, Howard B.; BEALE, Mark H.; DE JESÚS, Orlando. **Neural Network Design (2nd Edition)**. 2. ed. [*S.l.*]: Martin Hagan, 2014.

HAYKIN, Simon O. **Neural Networks and Learning Machines**. 3. ed. Upper Saddle River, NJ: Pearson, 2008. 936 p.

JOHNSON, Justin M.; KHOSHGOFTAAR, Taghi M. Survey on deep learning with class imbalance. **Journal of big data**, v. 6, n. 1, p. 54, mar. 2019.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, mai. 2015.

LI, Ya; TIAN, Xinmei; SHEN, Xu; TAO, Dacheng**. Classification and representation joint learning via deep networks**. *In***: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence**. California: International Joint Conferences on Artificial Intelligence Organization, 2017.

LIAO, Yiyi; KODAGODA, Sarath; WANG, Yue; SHI, Lei; LIU, Yong**. Understand scene categories by objects**. *In***: International Conference on Robotics and Automation (ICRA)**. [*S.l.*]: IEEE, 2016. p. 2215–2221.

LIU, Cheng-Lin; NAKASHIMA, Kazuki; SAKO, Hiroshi; FUJISAWA, Hiromichi. Handwritten digit recognition. **Pattern recognition**, v. 36, n. 10, p. 2271–2285, 2003.

MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **The Bulletin of Mathematical Biophysics**, v. 5, n. 4, p. 115–133, 1943.

MINSKY, Marvin. **Semantic Information Processing**. Cambridge, Mass.: MIT Press, 1968. 450 p.

MOMENY, Mohammad; LATIF, Ali Mohammad; AGHA SARRAM, Mehdi; SHEIKHPOUR, Razieh; ZHANG, Yu Dong. A noise robust convolutional neural network for image classification. **Results in engineering**, v. 10, p. 100225, 2021.

NAZARÉ, Tiago S.; COSTA, Gabriel B. Paranhos da; CONTATO, Welinton A.; PONTI, Moacir. Deep convolutional neural networks and noisy images. *In***: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications**. [*S.l.*]: Springer International Publishing, fev. 2018. v. 10657, p. 416–424.

NIELSEN, Michael A. **Neural Networks and Deep Learning**. [*S.l.*]: Determination Press, 2015.

NILSSON, Nils J. **Principles of artificial intelligence**. Oxford, England: Morgan Kaufmann, 1980. 476 p.

PRINCE, Simon J. D. **Computer vision**. 1. ed. Cambridge, England: Cambridge University Press, 2012. 598 p.

ROSENBLATT, Frank. The Perceptron, a Perceiving and Recognizing Automaton. *In***: Report: Cornell Aeronautical Laboratory**. [*S.l.*]: Cornell Aeronautical Laboratory, 1957.

ROY, Sudipta Singha; AHMED, Mahtab; AKHAND, Muhammad Aminul Haque**. Classification of massive noisy image using auto-encoders and convolutional neural network**. *In***: International Conference on Information Technology (ICIT)**. [*S.l.*]: IEEE, out. 2017. p. 971–979.

ROY, Sudipta Singha; AHMED, Mahtab; AKHAND, Muhammad Aminul Haque. Noisy image classification using hybrid deep learning methods. **Journal of Information and Communication Technology**, v. 17, n. 2, abr. 2018.

RUMELHART, David E.; MCCLELLAND, James L. **Parallel distributed processing**. London, England: MIT Press, 1986.

RUSSELL, Stuart; NORVIG, Peter. **Artificial intelligence**. 4. ed. London, England: Pearson Education, 2021. 1166 p.

SARKER, Iqbal H. Machine learning. **SN computer science**, v. 2, n. 3, mar. 2021.

SHINDE, Pramila P.; SHAH, Seema**. A review of machine learning and deep learning applications**. *In***: International Conference on Computing Communication Control and Automation (ICCUBEA)**. [*S.l.*]: IEEE, 2018.

SOLOMON, Chris; BRECKON, Toby. **Fundamentals of digital image processing**. Hoboken, NJ: Wiley-Blackwell, 2010. 335 p.

SZELISKI, Richard. **Computer Vision**. London, England: Springer, 2010. 832 p.

TEICHMANN, Marvin; WEBER, Michael; ZOLLNER, Marius; CIPOLLA, Roberto; URTASUN, Raquel**. MultiNet**. *In***: 2018 IEEE Intelligent Vehicles Symposium**. [*S.l.*]: IEEE, 2018. p. 1013–1020.

VINCENT, Pascal; LAROCHELLE, Hugo; BENGIO, Yoshua; MANZAGOL, Pierre-Antoine**. Extracting and Composing Robust Features with Denoising Autoencoders**. *In***: International Conference on Machine Learning**. Helsinki, Finland: Association for Computing Machinery, 2008. p. 1096–1103.

VINCENT, Pascal; LAROCHELLE, Hugo; LAJOIE, Isabelle; BENGIO, Yoshua; MANZAGOL, Pierre-Antoine. Stacked Denoising Autoencoders. **J. Mach. Learn. Res.**, JMLR.org, v. 11, p. 3371–3408, dez. 2010.

WAZLAWICK, Raul Sidnei. **Metodologia de Pesquisa para Ciência da Computação**. 3. ed. Rio de Janeiro, RJ: GEN LTC, 2020. 152 p.

WIDROW, Bernard; HOFF, Marcian E**. Adaptive switching circuits**. *In*: NEW YORK, 1**. IRE WESCON Convention Record**. [*S.l.*: *s.n.*], 1960. v. 4, p. 96–104.

WINSTON, Patrick Henry. **Artificial Intelligence**. 3. ed. Upper Saddle River, NJ: Pearson, 1992. 737 p.

XU, Qiang; ZENG, Yu; TANG, Wenjun; PENG, Wei; XIA, Tingwei; LI, Zongrun; TENG, Fei; LI, Weihong; GUO, Jinhong. Multi-Task Joint learning model for segmenting and classifying tongue images using a deep neural network. **Journal of biomedical and health informatics**, v. 24, n. 9, p. 2481–2489, 2020. IEEE.

ZHOU, Yiren; SONG, Sibo; CHEUNG, Ngai-Man**. On classification of distorted images with deep convolutional neural networks**. *In***: International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [*S.l.*]: IEEE, 2017.