Centro Federal de Educação Tecnológica de Minas Gerais
Engenharia Mecatrônica

Desenvolvimento de uma plataforma para identificação de deformidades de produtos em linhas de produção empregando técnicas de *machine learning* e redes neurais artificiais

Marcony Montini de Oliveira Lima

Divinópolis - 2018

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

CAMPUS DIVINÓPOLIS

GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

Marcony Montini de Oliveira Lima

DESENVOLVIMENTO DE UMA PLATAFORMA PARA IDENTIFICAÇÃO DE DEFORMIDADES DE PRODUTOS EM LINHAS DE PRODUÇÃO EMPREGANDO TÉCNICAS DE *MACHINE LEARNING* E REDES NEURAIS ARTIFICIAIS

Divinópolis.

2018.

Marcony Montini de Oliveira Lima

DESENVOLVIMENTO DE UMA PLATAFORMA PARA IDENTIFICAÇÃO DE DEFORMIDADES DE PRODUTOS EM LINHAS DE PRODUÇÃO EMPREGANDO TÉCNICAS DE *MACHINE LEARNING* E REDES NEURAIS ARTIFICIAIS

Trabalho de Conclusão de Curso apresentado ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.

Áreas de integração: Computação e Controle.

Orientador: Prof. Doutor Thiago Magela Rodrigues Dias

Divinópolis.

2018.

(Catalogação - Biblioteca Universitária – Campus Divinópolis – CEFET-MG)

L732d Lima, Marcony Montini de Oliveira.
Desenvolvimento de uma plataforma para identificação de deformidades de produtos em linhas de produção empregando técnicas de *machine learning* e redes neurais artificiais. / Marcony Montini de Oliveira Lima. – Divinópolis, 2018.
78f. : il.

Orientador: Prof. Dr. Thiago Magela Rodrigues Dias.

Trabalho de Conclusão de Curso (graduação) – Colegiado de Graduação em Engenharia Mecatrônica do Centro Federal de Educação Tecnológica de Minas Gerais.

1. Mecatrônica. 2. Computação. 3. Controle. 4. Redes Neurais Artificiais. 5. Deformidade. 6. *Machine Learning*. I. Dias, Thiago Magela Rodrigues. II. Centro Federal de Educação Tecnológica de Minas Gerais. III. Título.

CDU: 62(043)

Centro Federal de Educação Tecnológica de Minas Gerais
CEFET-MG / Campus Divinópolis
Curso de Engenharia Mecatrônica

Monografia intitulada "*Desenvolvimento de uma plataforma para identificação de deformidades de produtos em linhas de produção empregando técnicas de machine learning e redes neurais artificiais*", de autoria do graduando Marcony Montini de Oliveira Lima, aprovada pela banca examinadora constituída pelos seguintes professores:

_____

Prof. Doutor Thiago Magela Rodrigues Dias - CEFET-MG / Campus Divinópolis - Orientador

_____

Prof. Me. Lucas Silva de Oliveira - CEFET-MG / Campus Divinópolis

_____

Prof. Me. Eduardo Habib Bechelane Maia - CEFET-MG / Campus Divinópolis

_____

Prof. Dr. Lucio Flavio Santos Patricio
Coordenador do Curso de Engenharia Mecatrônica
CEFET-MG / Campus Divinópolis

Divinópolis - Dezembro de 2018

AGRADECIMENTOS

Agradeço, primeiramente, à Deus, por todas as oportunidades e caminhos que me foram apresentados no decorrer desta jornada. Em segundo, agradeço aos meus pais, Geovani e Lucelena, aos meus irmãos, Mayron e Muryel e aos meus avós, Maria Helena, Wanda e Olentino, pelas palavras de motivação, pela preocupação e pela ajuda fornecida, quando foi necessário. Agradeço também aos meus tios e tias, Fernanda, Sulamita, Luciano e Simone, e seus cônjuges.

Agradeço aos meus amigos de longa data, Juan, Wesley, Flávio, Leonardo, Thiago e Mateus. Amigos estes que também sofreram da mesma forma que eu: algumas noites mal dormidas, preocupação com aquela nota que não sai, aquele trabalho que não funciona... Apesar de alguns estarem distantes, o mal é o mesmo quando se trata de uma universidade federal. Também agradeço à Bah e Larissa, que apesar da distância, aturaram um pouco das minhas lamúrias.

Agradeço aos meus colegas das turmas de 2013, 2014 e 2015. Destaque para aqueles que se fizeram mais presentes: Cláudya, Keanu, Wanderson, Pedro Túlio, Giulia, Ferreira, Parreiras, Almeida, Vitória, Kesley, Delgado, Lanna, Rafael, Julie, Dani e Romulo.

Agradeço também aos professores, que tiveram papéis importantes durante este trajeto, mestres que nos ensinaram como fazer o certo que, além de educadores, eram nossos amigos, se mostrando preocupados quanto a nós. Em especial, aqueles se fizeram mais presentes, os professores Luiz Cláudio (Teacher), Rodrigo, Evandro Fockink, Jean, Lúcio, Emerson e Renato. Agradecimentos também ao professor Paganotti, por ter fornecido à mim o acesso ao laboratório para estudos e desenvolvimento do projeto.

Agradeço ao meu orientador, Prof. Doutor Thiago, por ter aceitado me orientar, mesmo que eu não tivesse noção nenhuma do assunto no começo do projeto, e aos senhores da banca, Prof. Me. Lucas e Prof. Me. Eduardo Habib, por terem aceito a proposta e terem contribuído com ideias e sugestões no desenvolvimento da mesma.

Agradeço ao CEFET, como instituição, e à todos que eu tenha esquecido de mencionar aqui, mas que contribuíram de alguma forma nesta caminhada. E, por fim, mas não menos importante, agradeço à própria sorte, que se fez presente em momentos cruciais nestes últimos anos.

RESUMO

A proposta do presente projeto foi o desenvolvimento de uma plataforma para análise de deformidades, contemplando as áreas da Computação e Controle. Teve-se por objetivo apresentar uma solução para problemas em inspeção de produtos em linhas de produção, uma vez que, em alguns casos, o trabalho manual é complexo, demorado ou repetitivo. A identificação das principais deformações apresentadas por um produto em sua linha de produção foi essencial para o treinamento da plataforma, efetuado através do processamento e análise de imagens, utilizando métodos de redes neurais artificiais e *machine learning*. Trabalhando com a simplificação de imagens, o estudo das deformações e suas características é necessário para melhor desempenho do algoritmo. Para a aquisição de dados, durante as etapas finais do trabalho, foi utilizada a câmera e componente de iluminação de um *smartphone* com resolução de 4160x3120 pixels, uma vez que certas condições devem ser exigidas para o processamento das imagens. Com o desenvolvimento de algoritmos, empregando a linguagem de programação *Python*, o objetivo deste trabalho foi de apresentar uma resposta rápida para o processo, sem interrupções à linha de produção, gerando uma melhor uniformidade e aumentando o rendimento na produção. O emprego de materiais para análise simples, como embalagens de jogos e bonecos, nas fases iniciais de teste, contribuiu para o reconhecimento e tratamento de características que dificultam a obtenção de resultados satisfatórios. Por fim, foi definido como objeto de análise ampolas/refis de insulina para canetas de aplicação. A razão desta escolha se deu por este material ser de fácil manuseio e de alta aplicação na área farmacêutica, visando verificar a funcionalidade da plataforma para identificação de variações de cores e volume do fluido. Neste projeto, o emprego de reconhecimento de padrões por redes neurais artificiais e *machine learning*, contempla a área da Computação, enquanto que o processamento das imagens, que está diretamente relacionado ao processamento de sinais, abrange a área de Controle. Assim, com este projeto, é apresentada uma plataforma que possa servir de opção na inspeção de produtos em uma linha de produção, atendendo a demanda de mercado e reduzindo custos desnecessários, tanto financeiros quanto temporais. Por fim, é importante ressaltar que foram obtidos resultados satisfatórios com valores interessantes de acurácia, porém, não foram realizados testes de aplicação em tempo real.

Palavras-chave: *Machine learning*, redes neurais artificiais, análise de deformidade.

ABSTRACT

The proposal of the present project was the development of a platform for analysis of deformities, contemplating the areas of Computation and Control. The objective was to present a solution to problems in product inspection on production lines, since in some cases manual labor is complex, time-consuming or repetitive. The identification of the main deformations presented by a product in its production line was essential for the training of the platform, carried out through the processing and analysis of images, using methods of artificial neural networks and machine learning. Working with the simplification of images, the study of the deformations and their characteristics is necessary for better performance of the algorithm. For the acquisition of data, during the final stages of the work, the camera and lighting component of a smartphone with a resolution of 4160x3120 pixels were used, since certain conditions must be required for the image processing. With the development of algorithms, using the Python programming language, the objective of this work was to present a rapid response to the process, without interruptions to the production line, generating a better uniformity and increasing yield in production. The use of simple materials for analysis, such as packaging of games and dolls, in the initial stages of testing, contributed to the recognition and treatment of characteristics that make difficult to obtain satisfactory results. Finally, it was defined as object of analysis ampoules / refills of insulin for application pens. The reason for this choice was that this material was easy to handle and of high application in the pharmaceutical area, aiming to verify the functionality of the platform to identify color variations and fluid volume. In this project, the use of pattern recognition by artificial neural networks and machine learning, encompasses the area of Computation, while the image processing, which is directly related to signal processing, covers the Control area. Thus, with this project, a platform is presented that can serve as an option in the inspection of products in a production line, meeting the market demand and reducing unnecessary costs, both financial and temporal. Finally, it is important to note that satisfactory results were obtained with interesting values of accuracy, however, no real-time application tests were performed.

Keywords: Machine learning, artificial neural networks, deformity analysis.

LISTA DE FIGURAS

Figura 1: Identificação de cartas de baralho pelo TensorFlow™.....10
Figura 2: Representação de uma imagem digital.....11
Figura 3: Variação da tonalidade em Grayscale de acordo com seu valor.....11
Figura 4: Representação dos canais RGB de uma imagem.....12
Figura 5: Representação do sistema de coordenadas em uma imagem.....12
Figura 6: Exemplificação de níveis de correlação entre classes.....14
Figura 7: Exemplificação da discriminação de classes através de atributos.....15
Figura 8: Exemplificação da organização das classes após processo PCA e whitening.....16
Figura 9: Exemplificação real da organização das classes após processo PCA e whitening.....16
Figura 10: Imagem em Grayscale após processo de binarização por limiar.....17
Figura 11: Exemplificação do processo de deteccção de bordas.....18
Figura 12: Decomposição em valores singulares de uma matriz A.....18
Figura 13: Hiperplano traçado entre dados de duas classes.....21
Figura 14: Exemplificação de classificadores lineares e não lineares.....22
Figura 15: Estrutura celular.....23
Figura 16: Parâmetros e seguimento de uma rede neural.....24
Figura 17: Perceptron de uma camada.....24
Figura 18: Rede neural artificial multicamada com uma camada oculta.....26
Figura 19: Exemplos de função de ativação.....27

Figura 20: Fluxograma do funcionamento de uma rede neural feedforward multicamada.....28
Figura 21: Exemplificação da descida do gradiente.....28
Figura 22: Exemplificação da descida do gradiente.....29
Figura 23: Exemplificação da diferença entre mínimo local e mínimo global.....29
Figura 24: Fluxograma para realização da descida do gradiente.....30
Figura 25: Relação feedforward e backpropagation em uma rede neural artificial.....31
Figura 26: Rede neural multicamada com quatro camadas ocultas e quatro neurônios na camada de saída.....33
Figura 27: Exemplificação do banco de dados MNIST.....34
Figura 28: Exemplos de imagens empregadas para classificação da caneta.....37
Figura 29: Exemplos de imagens empregadas para classificação da embalagem do jogo ditigal.....37
Figura 30: Exemplos de imagens empregadas para classificação do interior da embalagem do jogo ditigal.....38
Figura 31: Exemplos de imagens empregadas para classificação dos bonecos.....39
Figura 32: Exemplos de imagens empregadas para classificação do boneco.....39
Figura 33: Janela inicial para seleção do algoritmo de classificação.....40
Figura 34: Janela para entrada de parâmetros para treinamento por RNA.....41
Figura 35: Fluxograma do tratamento de imagem para extração de caraterísticas.....42
Figura 36: Exemplo de tratamento de imagem, representando: (a) a imagem original colorida; (b) imagem após filtragem com filtro mediano com vizinhança de 3 pixels; (c) imagem convertida para níveis de cinza; (d) imagem binarizada; (e) imagem colorida segmentada.....43
Figura 37: Rede neural implementada para classificação dos dados após tratamento especificado.....44

Figura 38: Relação da acurácia entre os métodos SVM e RNA para o banco de dados MNIST com 21000 amostras no treinamento e 21000 no teste.....46
Figura 39: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados do interior da embalagem de jogo digital com 4 saídas.....48
Figura 40: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados do interior da embalagem de jogo digital com 2 saídas.....49
Figura 41: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados dos 4 bonecos distintos.....50
Figura 42: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados de um boneco.....51

LISTA DE QUADROS E TABELAS

Quadro 1: Relação de algoritmos e acurácia ao longo da década.....9
Quadro 2: Funções de Kernel mais utilizadas no emprego de máquinas de vetor de suporte.....22
Quadro 3: Relação de taxa de erro para algoritmos aplicados ao MNIST.....35

Tabela 1: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados MNIST.....45
Tabela 2: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados de uma caneta.....47
Tabela 3: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados de imagens da embalagem de um jogo digital.....48
Tabela 4: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados do interior da embalagem de jogo digital estando vazia ou cheia.....50
Tabela 5: Relação de amostras nos bancos de dados.....52
Tabela 6: Relação de acurácia (%) no treinamento por RNA e tratamento SVD para as características de coloração e volume (próprio autor).....53
Tabela 7: Relação de acurácia (%) no treinamento por RNA e tratamento PCA para as características de coloração e volume (próprio autor).....54
Tabela 8: Relação de acurácia (%) no treinamento por RNA e tratamento de segmentação para as características de coloração e volume.....54
Tabela 9: Relação de acurácia (%) no treinamento por SVM e tratamento SVD para as características de coloração e volume.....55
Tabela 10: Relação de acurácia (%) no treinamento por SVM e tratamento PCA para as características de coloração e volume.....55
Tabela 11: Relação de acurácia (%) no treinamento por SVM e tratamento de segmentação para as características de coloração e volume.....56

LISTA DE ABREVIATURAS E SIGLAS

AM – Aprendizagem de Máquina

GUI – *Graphical User Interface*

PCA – *Principal Component Analysis*

SVD – *Singular-Value Decomposition*

SVM – *Support Vector Machine*

RNA – Rede Neural Artificial

SUMÁRIO

AGRADECIMENTOS vi

RESUMO vii

ABSTRACT viii

LISTA DE FIGURAS ix

LISTA DE QUADROS E TABELAS xii

LISTA DE ABREVIATURAS E SIGLAS xiii

SUMÁRIO xiv

1- INTRODUÇÃO 2

1.1- DEFINIÇÃO DO PROBLEMA 3

1.2- MOTIVAÇÃO 3

1.3- OBJETIVO GERAL 3

1.4- OBJETIVOS ESPECÍFICOS 4

1.5- METODOLOGIA 4

1.6- ESCOPO DO TRABALHO 6

2- FUNDAMENTAÇÃO 7

2.1- REVISÃO DA LITERATURA 7

2.2- ESTADO DA ARTE 8

2.3- FUNDAMENTAÇÃO TEÓRICA 10

2.3.1- AQUISIÇÃO DE DADOS 10

2.3.2- PRÉ-PROCESSAMENTO DE DADOS 13

2.3.2.1- NORMALIZAÇÃO 13

2.3.2.2- REDIMENSIONAMENTO 13

2.3.2.3- *GRAYSCALE* 13

2.3.2.4- PCA e *WHITENING* 14

2.3.2.5- BINARIZAÇÃO COM LIMIAR (THRESHOLDING) 17

2.3.2.6- SEGMENTAÇÃO E MÉTODOS DE DETECÇÃO DE BORDA 17

2.3.2.7- SVD (*SINGULAR VALUE DECOMPOSITION*) 18
2.3.2.8 – FILTRO DE MEDIANA 19
2.3.3- CLASSIFICAÇÃO E ANÁLISE.....19
2.3.3.1- *SUPPORT VECTOR MACHINES* 20
2.3.3.2- REDES NEURAIS ARTIFICIAIS 23
3- DESENVOLVIMENTO 34
4- RESULTADOS E DISCUSSÕES.....45
5- CONCLUSÕES.....59
5.1- PROPOSTAS PARA TRABALHOS FUTUROS 59
6- REFERÊNCIAS BIBLIOGRÁFICAS.....60

1- INTRODUÇÃO

O processo de automação tem crescido cada vez mais na área industrial, uma vez que proporciona menor tempo para se realizar uma tarefa, maior rendimento e melhor qualidade, quando empregado de forma correta, se comparado ao serviço braçal (Araújo Júnior, et al. 2003). Em uma linha de produção, automatizada ou não, existe a possibilidade do produto apresentar deformidades ou irregularidades, podendo estas terem sido originadas no processo de transporte ou transformação da matéria-prima empregada, antes mesmo de chegar na linha de montagem. A verificação destas inconformidades é, em alguns casos, realizada de forma visual, onde um funcionário acompanha o trajeto dos produtos.

Diversos sistemas de automação e controle têm sido desenvolvidos a fim de otimizar o processo em uma linha de produção. O alto desempenho e processamento dos atuais sistemas eletroeletrônicos possibilitam uma excelente performance na solução de problemas reais através de respostas rápidas. Quando um produto e/ou serviço não atendem às especificações de qualidade do cliente, existe uma perda no processo de fabricação. Estabelecer a diferença entre verificar para “prevenir” produtos defeituosos e verificar para “localizar” defeitos é essencial para se reduzir estas perdas (Abreu, 2002).

Existe uma ligação íntima entre a questão de qualidade e a análise das perdas para sua eliminação. Reduzindo estas perdas, é possível gerar recursos que poderão ser aplicados em formas de alavancar o sistema de qualidade, trazendo benefícios que, futuramente, serão capazes de suplantar os gastos iniciais (Robles, 1994).

Logo, este trabalho almeja o desenvolvimento de uma plataforma que atue na detecção de deformidades em peças ou produtos em uma linha de produção, sendo estas deformidades superficiais, uma vez que a análise é realizada a partir da imagem obtida por câmeras, utilizadas para este propósito. A classificação das imagens é efetuada a partir de técnicas de redes neurais artificiais e *machine learning*. Ao se identificar a presença de irregularidades no produto, o mesmo pode ser retirado da linha de montagem e separado dos demais através da aplicação de um atuador simples, por exemplo.

Com a plataforma proposta, espera-se apresentar uma opção de ferramenta para manter a uniformidade na linha de produção. Supondo-se que

seja implementado um canal de comunicação com um atuador que possa retirar o produto danificado da linha no momento de sua identificação, sem gerar atrasos durante a análise dos produtos, pode ser possível também exprimir os tipos de deformidades mais comuns encontrados e resultar em uma concebível solução para suprimi-las. Não é o objetivo deste trabalho aplicar a conexão à um atuador, mas é um possível cenário de aplicabilidade à plataforma. Tais processos, consequentemente, gerariam uma redução de custos em materiais que seriam descartados.

Sendo assim, o trabalho contemplará duas das grandes áreas da Engenharia Mecatrônica: Computação e Controle. Foi realizado o desenvolvimento de uma plataforma para aplicação de técnicas de redes neurais artificais e *machine learning* e, também, o processamento de imagens, que está diretamente relacionado ao processamento de sinais, uma vez que, os sinais, como as imagens digitais, são na realidade um suporte físico que carrega no seu interior uma determinada informação (Portes de Albuquerque, 2000).

1.1- DEFINIÇÃO DO PROBLEMA

O problema de perda de tempo e, consequentemente, financeira, buscando melhorar o processo de identificação de deformidades em produtos em uma linha de produção.

1.2- MOTIVAÇÃO

A motivação para a realização deste trabalho surgiu do grande interesse do autor na área de Computação, além da vontade de aprofundamento em temas como visão computacional e redes neurais artificiais.

1.3- OBJETIVO GERAL

Desenvolver uma plataforma para análise de deformidades empregando visão computacional e técnicas de classificação de imagens que mantenha a uniformidade em uma linha de produção e ajude a reduzir gastos.

1.4- OBJETIVOS ESPECÍFICOS

- Implementar técnicas básicas de redes neurais artificiais e *machine learning* para classificação de objetos;
- Realizar o treinamento para reconhecimento e classificação de diversos tipos de deformidades físicas em um determinado objeto;
- Realizar o treinamento para reconhecimento e classificação de diversos tipos de deformidades físicas em diferentes objetos;
- Desenvolver um *software* que trabalhe com visão computacional e aplicar técnicas de pré-processamento de imagens;
- Implementar classificação de texturas para análise de deformidades superficiais.

1.5- METODOLOGIA

Este projeto tem como objetivo apresentar uma plataforma para inspeção de objetos ou peças em uma linha de produção. A aquisição de dados a serem analisados é realizada através de câmeras que, como componentes de sistema de aquisição, possuem uma matriz de sensores sensíveis a luz (Stivanello, 2004). Com a aplicação de luz sobre o objeto de análise, a sua reflexão incide sobre a câmera, gerando um valor de intensidade em cada coordenada da matriz, formando uma imagem digital. Para tal, foi utilizada a câmera e componente de iluminação de um smartphone com resolução de 4160x3120 pixels.

A técnica de iluminação a ser empregada varia de acordo com o resultado a ser buscado, ressaltando diferentes características do produto. Utilizando um anel luminoso em torno da câmera, por exemplo, é possível trabalhar com reconhecimento de caracteres. Outro exemplo é a iluminação por trás do produto, que proporciona uma imagem destacando sua silhueta, útil para análise de suas bordas.

Na etapa de processamento de imagens, são estudadas, visando a possibilidade de aplicação, diversas técnicas como:

- a representação de imagens monocromáticas, que utiliza escalas de cinza de 0 a 255 em uma matriz, onde cada coordenada representa um ponto específico da imagem;

- pré-processamento, melhorando a qualidade da imagem aplicando realce e restauração;
- segmentação de imagens para isolar os objetos de interesse na imagem;
- limiarização, onde a imagem é transformada em binária a partir de um valor de cinza específico, em que um algoritmo transforma valores acima deste em 255 e abaixo em 0;
- detecção de bordas, que é uma técnica em que a segmentação é realizada com base na descontinuidade de valores de níveis de cinza (Stivanello, 2004);
- descritores de imagem, que simplificam a área da imagem estudada, podendo ser estabelecidos pelo método do código de cadeia ou por descritores de Fourier;
- interpretação de imagens, para análise de padrões a partir das etapas de processamento anteriores.

Em se tratando da aplicação de redes neurais artificiais e *machine learning*, o objetivo é encontrar e classificar padrões, além de generalizar as informações, através da linguagem de programação *Python*.

Para o desenvolvimento da rede neural artificial, primeiramente, é realizada a sua definição, considerando seu tamanho, o tipo de problema a ser resolvido e o tipo de aprendizado do algoritmo. Em seguida, inicia-se a etapa de treinamento supervisionado, em que são apresentados pares de situações onde, para cada entrada, a saída desejada também seja mostrada. Se a saída obtida se difere da desejada, é realizado um “aprendizado” da rede. Se a saída obtida for igual à desejada, é apresentado um novo par. Este procedimento é repetido diversas vezes, aumentando a probabilidade de acerto do algoritmo. Por fim, é realizada sua implementação.

Se tratando das técnicas de aprendizagem de máquina, escolheu-se utilizar máquinas de vetores de suporte (*support vector machines*). Esta técnica trata de classificar cada uma das amostras apresentadas na etapa de treinamento através de separação linear. Com base nesta separação, é realizada uma comparação entre as amostras de treinamento e de teste, gerando uma classificação.

O projeto foi dividido em duas etapas. O procedimento metodológico a ser realizado na primeira etapa é a implementação de ambas as técnicas (*support*

vector machines e redes neurais artificiais) para classificação de objetos do cotidiano. Com o resultado de cada processo, foi realizada a análise e comparação de eficiência para definição de restrições e particularidades em relação a cada teste proposto. Também, a partir destes resultados, foram averiguadas as principais deficiências, tanto do código quanto dos métodos empregados, buscando melhorias para a segunda etapa, onde foi realizada a identificação de um objeto de análise específico: ampolas de refil de canetas para aplicação de insulina do tipo humana regular.

Como prova de conceito, foi utilizado um refil vazio do medicamento NovoRapid®. NovoRapid® é uma insulina moderna (insulina análoga) de ação ultra-rápida. Insulinas modernas são versões aprimoradas da insulina humana. Diabetes mellitus é uma doença na qual seu organismo não produz insulina suficiente para controlar o nível de açúcar no sangue. Este medicamento é usado para tratar diabetes mellitus em adultos, adolescentes e crianças (a partir de 2 anos de idade) (NovoRapid® FlexPen®, 2017). Uma das formas de aplicação é através de uma caneta, com uma agulha na ponta, em que o medicamento é adicionado à caneta através de um refil. Para treinamento e testes, utilizou-se imagens do refil contendo água, leite, uma mistura turva de água com leite, além de fluidos com coloração turva vermelha e marrom, variando o volume.

1.6- ESCOPO DO TRABALHO

Se tratando do escopo deste trabalho, primeiramente é apresentado o capítulo de fundamentação, composto pela revisão da literatura, metodologia e fundamentação teórica, apresentando toda a teoria estudada.

Em seguida, é denotado o capítulo de desenvolvimento, que trata de todos os processos e testes, tal como sua realização.

Posteriormente, o capítulo de resultados e discussões acerca de todo o trabalho desenvolvido.

Por fim, um capítulo apresentando propostas para trabalhos futuros e um capítulo contemplando as referências bibliográficas.

2- FUNDAMENTAÇÃO

Neste capítulo é tratado todo o embasamento teórico e histórico acerca do assunto, tal como os métodos a serem empregados para sua realização.

2.1- REVISÃO DA LITERATURA

As primeiras informações mencionadas sobre redes neurais artificiais são datadas de 1943, em que foi sugerida a construção de uma máquina ou dispositivo baseado no cérebro humano, em artigos de Warren McCulloch e Walter Pitts. Em 1949, Donald Hebb escreveu "*The Organization of Behavior*" (A Organização do Comportamento), um livro que defendia a ideia de que o condicionamento psicológico clássico é uma propriedade de neurônios individuais, logo, está presente em qualquer parte dos animais. Esta concepção atuou como inspiração para diversos outros pesquisadores da época (Bocanegra, et al. 2002).

Em 1951, foi construído, por Mavin Minsky, o primeiro neurocomputador, denominado Snark. Ajustando seus pesos automaticamente, por meio de um ponto de partida técnico, este computador operava perfeitamente, porém nunca efetuou alguma operação de processamento de informação interessante. Já em 1957 e 1958 surgiu o primeiro neurocomputador a obter sucesso, denominado Mark I Perceptron, e criado por Frank Rosenblatt, Charles Wightman e outros. Em seguida, foi desenvolvido, por Bernard Widrow, com ajuda de alguns estudantes, um novo tipo de elemento de processamento de redes neurais, denominado Adaline (Bocanegra, et al. 2002).

John Hopfield, renomado físico de reputação mundial, se interessou pela área e escreveu diversos artigos, persuadindo diversos pesquisadores, cientistas, matemáticos e tecnólogos a se unirem. Em 1986, editado por David Rumelhart e James McClelland, foi publicado o livro "*Parallel Distributed Processing*" (Processamento Distribuído Paralelo), trazendo mais pessoas à área da neurocomputação (Bocanegra, et al. 2002).

As primeiras Redes Neurais Convolutivas (*Convolutional Neural Networks*, CNNs) foram propostas em 1998 por LeCun, onde os autores desenvolveram uma arquitetura neural conhecida como LeNet5, empregada no reconhecimento de dígitos escritos à mão. Tal arquitetura estabeleceu um novo estado da arte, na ocasião, ao atingir acurácia de 99.2% na base de dados MNIST (LeCun, et al.

1998). Após alguns anos, Krizhevsky estabeleceu um marco na área ao propor a AlexNet, arquitetura vencedora do desafio ImageNet. A aplicação mais popular relacionada a Redes Neurais Convolutivas, desde então, tem sido o reconhecimento de padrões em imagens (Krizhevsky, et al. 2012).

Se tratando da ideia principal deste trabalho, que é a classificação de objetos para identificação de deformidades, alguns trabalhos acerca do assunto podem ser citados. Pesquisadores, em busca de uma melhoria de qualidade no setor alimentício, aplicaram redes neurais artificiais na análise de laranjas (Louro, 2006), maçãs (Nakano, 1997) e nozes (Ghazanfari, 1996), por exemplo.

2.2- ESTADO DA ARTE

Machine Learning (Aprendizagem de Máquina) trata-se de técnicas empregadas para realizar uma determinação ou predição baseada na utilização de algoritmos que coletam uma massiva quantidade de dados e aprendem com eles. Na última década, algoritmos de redes neurais artificiais têm se tornado mais comuns para este tipo de tarefa. Apesar de serem bastante semelhantes à aprendizagem de máquinas, sua estruturação e funcionamento baseiam-se na de um sistema neural real (Martins, et al. 2018).

O desenvolvimento de diversas técnicas e algoritmos empregando redes neurais artificais tem aumentado na última década. Como pode ser observado no Quadro 1, tal evolução tem proporcionado grande melhoria nos resultados. Neste quadro são apresentados algoritmos do estado da arte no reconhecimento de objetos ao longo dos anos de 2011 e 2014, na base CIFAR-10, e suas respectivas acurácias. Esta base de dados consiste de imagens coloridas de 32 x 32 pixels, divididas em 10 classes, com 50 mil imagens sendo aplicadas na fase de treino e 10 mil na fase de testes.

Quadro 1: Relação de algoritmos e acurácia ao longo da década (Rocha, et al. 2015)

Título	Autores	Acurácia
Spatially-Sparse Convolutional Neural Networks	(Graham, 2014)	93,72%
Networks in Networks	(Min Lin, et al. 2013)	91,19%
Maxout Networks	(Goodfellow, et al. 2013)	90,65%
Practical Bayesian Optimization of Machine Learning Algorithms	(Snoek, et al. 2012)	90,50%
ImageNet Classification with Deep Convolutional Neural Networks	(Krizshevsky, et al. 2012)	89%
Improving neural networks by preventing co-adaptation of feature detectors	(Hinton, et al. 2012)	84,40%
Learning Invariant Representation with Local Transformations	(Sohn, et al. 2012)	82,40%
Na Analysis of Single-Layer Networks in Unsupervised Feature Learning	(Coates, et al. 2011)	79,60%

Uma classe de rede neural artificial é a convolutiva. Apesar de sua estruturação complexa, pode-se utilizar Redes Neurais Convolutivas de maneira bem simples e prática, por meio do uso de *frameworks*. Sua utilização tem sido a escolha de usuários iniciantes, devido à simplicidade, bastando apenas escolher os parâmetros específicos para este propósito, inserir os dados de treinamento e executar, enquanto que outros necessitam de maior elaboração. Além disso, pode-se utilizar GPUs (*Graphics Processing Units*, ou Unidades de Processamento Gráfico) para um processamento mais rápido, dependendo da ferramenta utilizada e da disponibilidade gráfica na máquina.

O TensorFlow™ é um *framework* desenvolvido pela Google que é massivamente empregado na área de *machine learning* e *deep learning*, além de muitos outros domínios científicos. É uma biblioteca de código aberto para computação numérica de alto desempenho. Sua arquitetura flexível permite fácil implantação de computação em várias plataformas (CPUs, GPUs, TPUs) e de *desktops* em *clusters* de servidores para dispositivos móveis (Tensorflow, 2018). A Figura 1 apresenta a identificação de algumas cartas de baralho utilizando o TensorFlow™ e a porcentagem de certeza que o *software* possui para cada carta, após realizado o treinamento.

Figura 1: Identificação de cartas de baralho pelo TensorFlow™ (GitHub, 2018)

Outra opção de *framework* é o Keras, que nada mais é do que uma API (*Application Programming Interface*, ou Interface de Programação de Aplicações) de redes neurais artificiais de alto nível, escrita em Python (Keras, 2018).

Ambos os *frameworks*, Keras e TensorFlow, apresentam soluções para projetos em curta e ampla escala, porém demandam de uma noção prévia de programação e redes neurais artificiais para sua utilização. Em contrapartida, este trabalho propõe apresentar uma plataforma que realize a classificação de imagens, necessitando apenas que o usuário forneça os parâmetros desejados.

2.3- FUNDAMENTAÇÃO TEÓRICA

Esta seção trata de todo o embasamento teórico e conceitos estudados para o desenvolvimento deste projeto.

2.3.1- AQUISIÇÃO DE DADOS

A aquisição dos dados em um determinado sistema é fundamental para a realização de seu controle, uma vez que o monitoramento das variáveis que o influenciam é indispensável. O tipo de sensor a ser utilizado varia de acordo com o tipo de maquinário, podendo ser um sensor ultrassônico, de temperatura, de pH, entre outros, atuando através da transformação de estados físicos em valores numéricos. Para este projeto, foram empregadas técnicas de visão computacional, dado que uma imagem é vista pelo computador como uma matriz de números, conforme Figura 2.

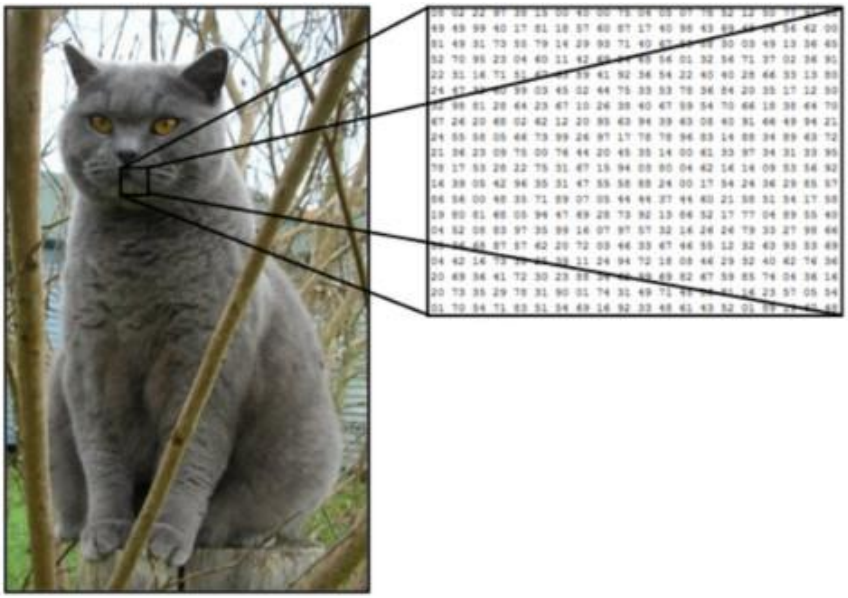

Figura 2: Representação de uma imagem digital (Ravindra, 2017)

A imagem digital colorida é uma matriz de 3 dimensões, contendo em cada um dos planos de profundidade um canal, sendo cada um destes canais uma das 3 cores do padrão RGB (red=vermelho, green=verde, blue=azul). Cada célula dessa matriz é um pixel, possuindo um valor de 0 a 255, sendo 0 para o mais escuro e 255 para o mais claro (Calixto, et al. 2016).

No caso de uma imagem preto e branca, também chamada de *Grayscale* (nível de cinza), temos apenas um canal, ou seja, apenas uma matriz de 2 dimensões. Portanto, cada célula contém um inteiro de 8 bits que, em *Python*, é definido por “uint8” que é um *unsigned integer* de 8 bits (Antonello, 2017). A Figura 3 apresenta a variância de níveis de cinza em uma imagem em *Grayscale* de acordo com o valor de cada pixel.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255
2		15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255
3		15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255
4		15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255
5		15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255
6		15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255
7		15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255
8		15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255
9		15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255
10		15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255

Figura 3: Variação da tonalidade em *Grayscale* de acordo com seu valor (Antonello, 2017)

Simplificando, para imagens coloridas temos três destas matrizes de duas dimensões, cada uma representando uma das cores do sistema RGB. As imagens coloridas, portanto, são compostas normalmente de 3 matrizes de inteiros de 8 bits. A junção das 3 matrizes, como pode ser observado na Figura 4, produz a imagem colorida com capacidade de reprodução de 16,7 milhões de cores, sendo que os 8 bits têm capacidade para 256 valores e, elevando a 3, temos 256^3 = 16,7 milhões (Antonello, 2017).

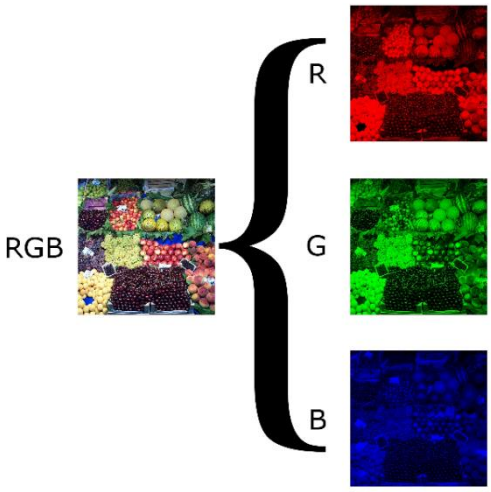

Figura 4: Representação dos canais RGB de uma imagem (Wikipedia: *Grayscale*)

Para a manipulação dos valores de cada pixel da imagem, é importante entender o sistema de coordenadas (linha, coluna), onde o pixel mais à esquerda e acima da imagem está na posição (0,0). Já em uma imagem com 400 pixels de largura e 300 pixels de altura, ou seja, 400 colunas e 300 linhas, o pixel (299,399) será o mais à direita e abaixo da imagem. Tomando a Figura 5 como exemplo, é possível observar que o pixel na coordenada (4,5) possui o valor 200.

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
L0	0	0	0	0	0	0	0	0	0	0
L1	0	50	50	50	50	50	50	50	50	0
L2	0	50	100	100	100	100	100	100	50	0
L3	0	50	100	150	150	150	150	100	50	0
L4	0	50	100	150	200	200	150	100	50	0
L5	0	50	100	150	200	200	150	100	50	0
L6	0	50	100	150	150	150	150	100	50	0
L7	0	50	100	100	100	100	100	100	50	0
L8	0	50	50	50	50	50	50	50	50	0
L9	0	0	0	0	0	0	0	0	0	0

Figura 5: Representação do sistema de coordenadas em uma imagem (Antonello, 2017)

2.3.2- PRÉ-PROCESSAMENTO DE DADOS

Algoritmos de aprendizado de máquina necessitam de uma grande quantidade de exemplos para funcionar bem. Uma hipótese somente pode ser útil se puder ser utilizada para reconhecer corretamente exemplos além dos utilizados na indução da hipótese. Portanto, deve-se ter cuidado ao induzir uma hipótese para que ela não seja excessivamente especializada aos exemplos utilizados para criá-la, que é um problema chamado de *overfitting* (Batista, 2003). Com este problema, o algoritmo perde a capacidade de generalização do caso, ao se adaptar a valores muito específicos do treino. Criar novos exemplos a partir de tratamentos de pré-processamento é uma opção para aumentar a eficácia do algoritmo e diminuir a chance de perda da habilidade de generalização.

2.3.2.1- NORMALIZAÇÃO

Consiste em transformar os valores dos atributos de seus intervalos originais para um intervalo específico, como, por exemplo, [-1, 1] ou [0, 1]. Esse tipo de transformação é particularmente valiosa para métodos que calculam distâncias entre atributos. Métodos como redes neurais artificiais possuem um rendimento melhor quando treinados com valores de atributos pequenos (Batista, 2003).

2.3.2.2- REDIMENSIONAMENTO

Uma técnica que mantém a estrutura do objeto na imagem, modificando a distância relativa dos pixels, tornando o modelo mais robusto. Simplificando, trata da modificação dos valores de altura e largura da imagem (Silva, et al. 2011).

2.3.2.3- *GRAYSCALE*

Como já citado, em *Grayscale* a imagem é definida apenas em uma matriz bidimensional, diferente do RGB, que é tridimensional (Nogueira, 2016). Ao realizar esta conversão de RGB para *Grayscale*, a imagem diminui para 1/3 do seu número total de variáveis, diminuindo o custo computacional no processamento.

2.3.2.4- PCA e *WHITENING*

No PCA, do inglês *Principal Component Analysis* (Análise de Componentes Principais), é feita uma transformação linear do dado de entrada x de d dimensões para m dimensões, de forma que m seja menor do que d e preserve o máximo de variância entre os dados. É importante que os atributos selecionados para descrever os objetos numa imagem sejam descorrelacionados. Para fins práticos, conforme Figura 6, a correlação é uma medida de quão bem se pode modelar a relação entre dois atributos através de uma função linear (Augusto, et al. 2012).

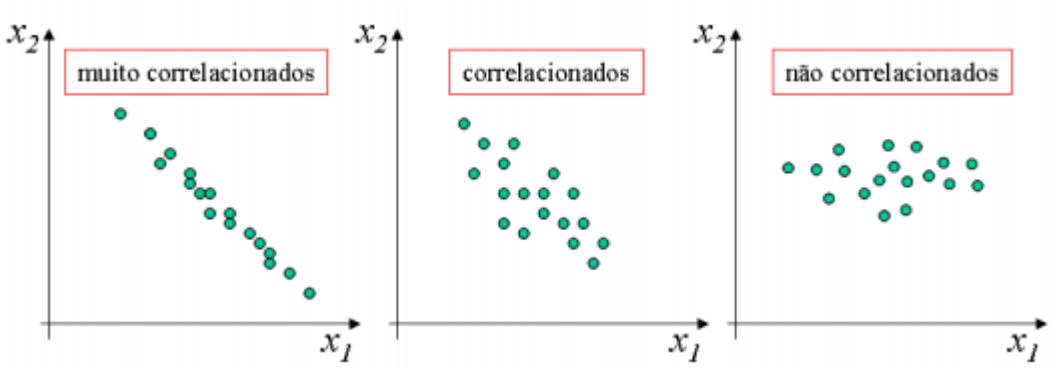

Figura 6: Exemplificação de níveis de correlação entre classes (Augusto, et al. 2012)

É necessário que poucos atributos sejam suficientes para discriminar os objetos de uma imagem, para facilitar o projeto e o treinamento do classificador. Assim, o PCA tem como objetivo reduzir o número de atributos por translação e rotação dos eixos, de modo a minimizar o erro de reconstrução. No exemplo da Figura 7, observa-se que não é possível separar as classes utilizando-se os dois atributos x_1 e x_2. Somente um novo atributo construído a partir de x_1 e x_2, a componente y_1, é capaz de discriminar as classes sozinho (Augusto, et al. 2012).

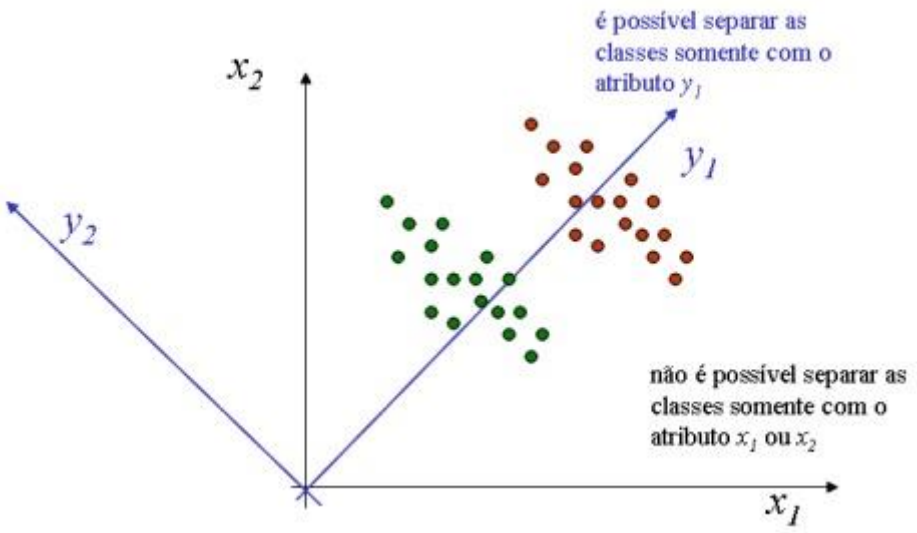

Figura 7: Exemplificação da discriminação de classes através de atributos (Augusto, et al. 2012)

$$C = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)(x_i - \mu)^T \quad (1)$$

$$Ce = \lambda e \quad (2)$$

Em que C é a matriz de covariância amostral, N é o número de padrões na amostra, x é o valor da amostra, μ é a média aritmética dada pela equação $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$, e é o autovetor e λ é o autovalor.

$$y = x_o P \quad (3)$$

Em que y é o valor do novo banco de dados, x_o é o valor do banco de dados inicial e P são os autovetores, dado por: $P = [e_1 e_2 \dots e_n]$ e $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

Algebricamente, o PCA é feito da seguinte forma (Santo, 2012):

• A base de dados é ajustada pela subtração da média correspondente a cada dimensão.

• É calculada a matriz de covariância, dada pela eq. (1).

• São calculados os autovalores e autovetores por meio da eq. (2), onde C foi calculado através da eq. (1). O autovetor associado ao maior autovalor é a componente mais relevante. Quanto maior o autovalor, maior a importância do autovetor.

• Para se obter a representação de cada padrão na base do PCA, constrói-se um novo conjunto de dados através da eq. (3), onde os autovetores componentes de P são dados a partir de e, definido na eq. (2).

A operação *whitening* (clareamento) toma os dados na autobase e divide todas as dimensões pelo autovalor para normalizar a escala. A interpretação geométrica dessa transformação é que, se os dados de entrada forem gaussianos multivariáveis, os dados embranquecidos serão gaussianos com média zero e matriz de covariância de identidade.

Verificando a Figura 8, é possível observar a organização das classes após o processo PCA (*decorrelated data*) e após o *whitening* (whitened data).

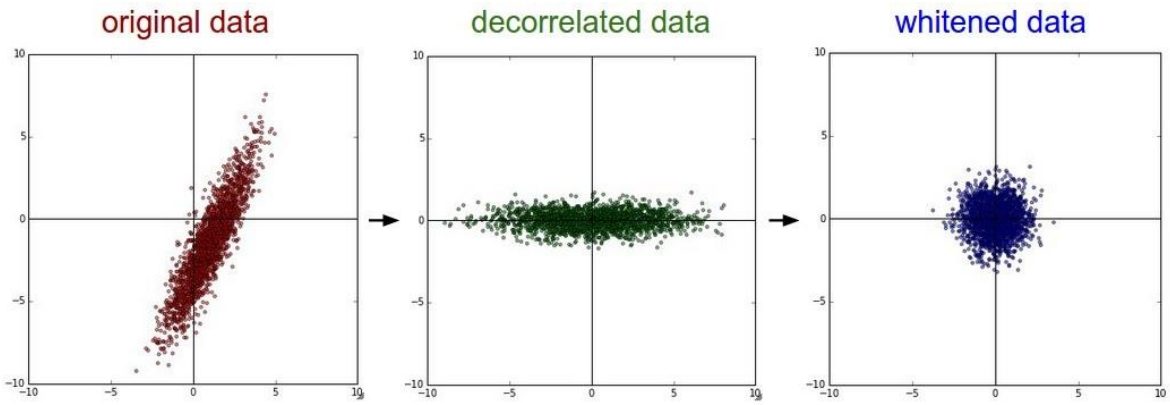

Figura 8: Exemplificação da organização das classes após processo PCA e *whitening* (GitHub, 2018)

Através da Figura 9, é possível ter uma visão realista de como as imagens ficam após o processo PCA (*reduced images*, usando os 144 autovetores associados aos maiores autovalores em *top 144 eigenvectors*) e *whitening* (whitened images).

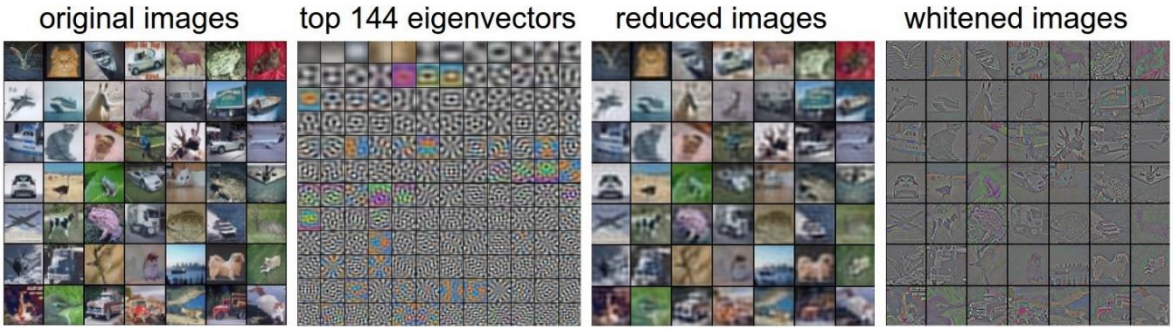

Figura 9: Exemplificação real da organização das classes após processo PCA e *whitening* (GitHub, 2018)

2.3.2.5- BINARIZAÇÃO COM LIMIAR (THRESHOLDING)

Thresholding, também traduzido como limiarização, é utilizado na maior parte das vezes, em processamento de imagens, para binarização da imagem. Normalmente, converte-se imagens em tons de cinza para imagens preto e branco onde todos os pixels possuem 0 ou 255 como valores de intensidade (Antonello, 2017). A Figura 10 exemplifica o resultado do processo.

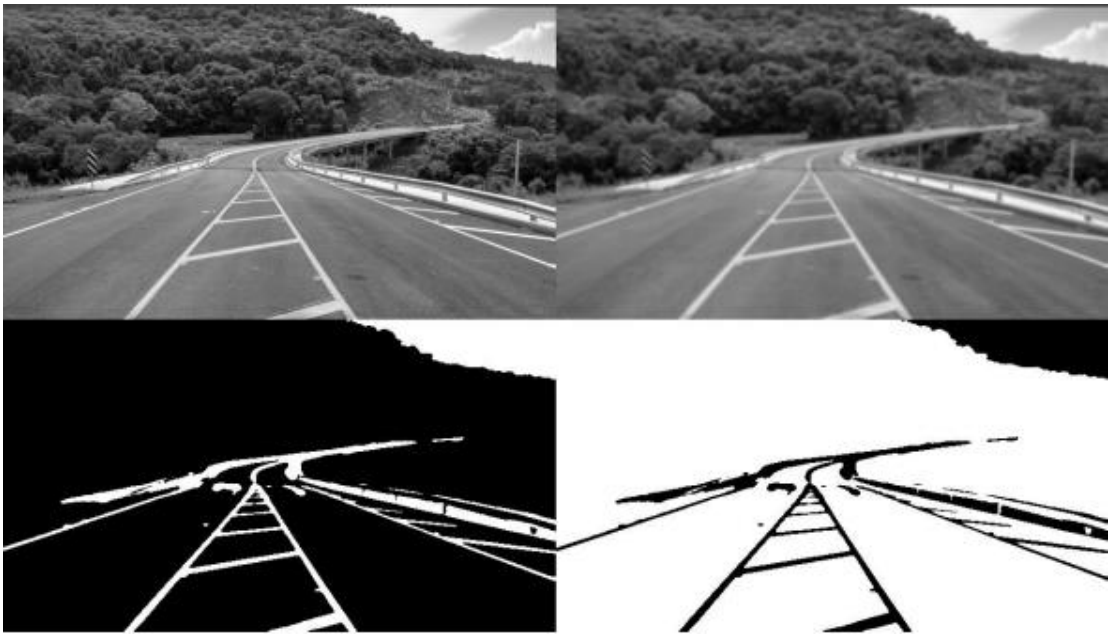

Figura 10: Imagem em *Grayscale* após processo de binarização por limiar (Antonello, 2017)

2.3.2.6- SEGMENTAÇÃO E MÉTODOS DE DETECÇÃO DE BORDA

Uma das tarefas mais importantes para a visão computacional é identificação de objetos. Para essa identificação uma das principais técnicas é a utilização de detectores de bordas, a fim de delimitar os formatos dos objetos presentes na imagem. Basicamente, em algoritmos para aplicação destas técnicas, a detecção de bordas se faz através da identificação de variações abruptas na intensidade dos pixels de uma região da imagem ou do gradiente (Maturana, 2010). A Figura 11 exemplifica o resultado deste processo.

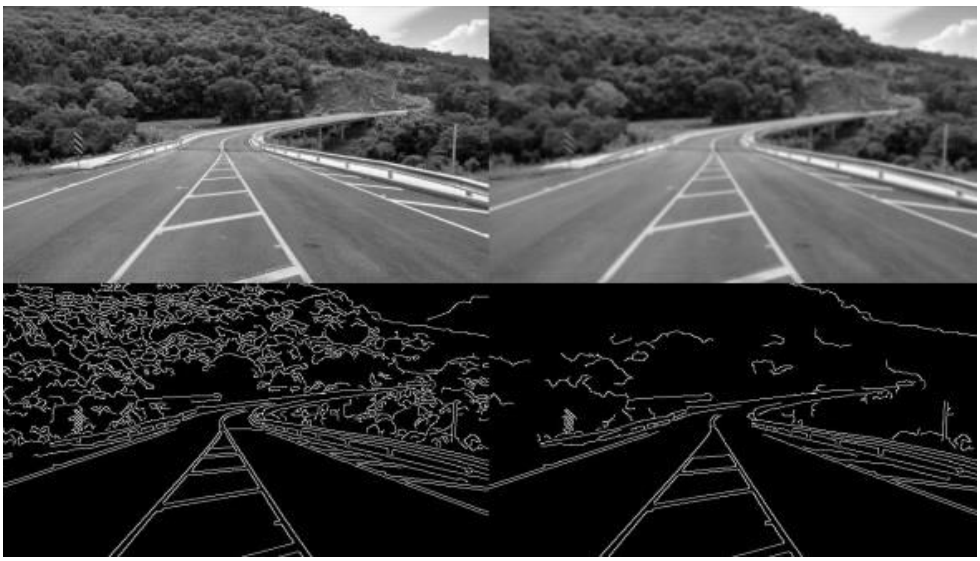

Figura 11: Exemplificação do processo de deteccção de bordas (Antonello, 2017)

2.3.2.7- SVD (*SINGULAR VALUE DECOMPOSITION*)

Este método trata da fatorização de uma matriz qualquer em três outras matrizes com características importantes. Desta forma, dado $A \in \mathbb{R}^{m \times n}$, a decomposição em valores singulares desta matriz A é uma fatorização $A = U\sum V^T$, conforme Figura 12, em que (Oliveira, 2016):

- $U \in \mathbb{R}^{m \times m}$ é ortogonal;
- $V \in \mathbb{R}^{n \times n}$ é ortogonal;
- $\sum \in \mathbb{R}^{m \times n}$ é diagonal se $m = n$, caso contrário adiciona-se $m - n$ linhas de zero em D.

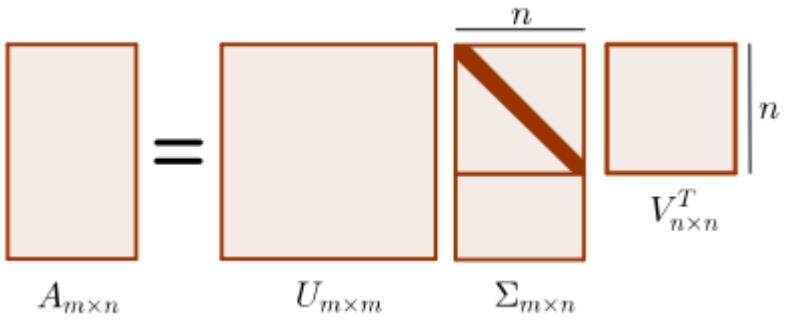

Figura 12: Decomposição em valores singulares de uma matriz A (Oliveira, 2016)

Assim, qualquer matriz $A_{n \times p}$ pode ser decomposta por valor singular na forma $A = U\sum V^T$, em que $U^T U = V^T V = VV^T = I_P$ e $\sum = diag(d_1, \ldots, d_p)$ com $d_1 \geq$

$d_2 \geq, \ldots, \geq d_p \geq 0$. As matrizes $A^T A$ e AA^T têm os mesmos autovalores e os elementos d_i são a raiz quadrada destes autovalores; a *i*-ésima coluna $v_i = (v_{i1}, \ldots, v_{ip})^T$ da matriz $V_{p\times p}$ é o autovetor correspondente ao *i*-ésimo maior autovalor d_i^2 de $A^T A$; enquanto a *j*-ésima coluna $u_j = (u_{1j}, \ldots, u_{nj})^T$ da matriz $U_{n\times p}$ é o autovetor correspondente ao *j*-ésimo maior autovalor d_j^2 de AA^T (Garcia-Pena, et al. 2014).

Os valores $\sigma_1, \sigma_2, \ldots, \sigma_n$ da diagonal principal de $\sum$ são chamados valores singulares de A e $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$. As colunas de $U, u_1, u_2, \ldots, u_m$, são chamadas de vetores singulares à esquerda de A enquanto que as colunas da matriz $V, v_1, v_2, \ldots, v_n$, são chamados de vetores singulares à direita de A de tal forma que $Av_j = \sigma_j u_j$ (Oliveira, 2016).

2.3.2.8 – FILTRO DE MEDIANA

Este filtro trabalha, retendo os detalhes da imagem, eliminando ruídos que não dependem dos valores que são significativamente diferentes dos valores típicos em uma vizinhança (Jain, et al. 1995). Um pixel é selecionado na imagem e é formado um conjunto, contendo o valor deste pixel e de seus vizinhos. Então, deste conjunto, é verificado o valor da mediana e, então, este valor é atribuído ao pixel selecionado. Por exemplo, na utilização de um filtro mediano com vizinhança de 3 pixels, são verificados os valores de cada pixel em ordem crescente e o quinto maior valor é atribuído ao pixel central da matriz 3x3 (Sanches, et al. 2015).

2.3.3- CLASSIFICAÇÃO E ANÁLISE

Nesta seção são tratados os métodos de Aprendizagem de Máquina e Redes Neurais Artificiais (RNAs) empregados para classificação das imagens.

Três paradigmas podem ser utilizados na geração de um preditor por meio de técnicas de Aprendizagem de Máquina e Redes Neurais Artificais: superviosionado, não-supervisionado e por reforço (Haykin, 1999). A escolha de um paradigma de aprendizado define o modo como ocorrerá o seu aprendizado por meio de um conjunto de dados, ou seja, a maneira como o algoritmo se relaciona com seu meio ambiente (Lorena, et al. 2003):

• Se tratando de aprendizado supervisionado, o algoritmo é treinado a partir de conjuntos de exemplos rotulados com o objetivo de aprender uma função desejada. É apresentada a figura de um “professor externo”, o qual fornece um conhecimento do ambiente representado por conjuntos de exemplos na forma entrada-saída.

• No paradigma de aprendizado não-supervisionado não existem instâncias rotuladas da função a ser aprendida, logo, não há a presença de um professor. O algoritmo aprende a representar (ou agrupar) as entradas submetidas segundo uma medida de qualidade.

• Já no paradigma de aprendizagem por reforço, o aprendizado se dá por meio de recompensas ou não ao indutor, conforme seu desempenho em aproximar a função desejada.

Neste projeto foram empregadas técnicas de aprendizado supervisionado, apresentando ao algoritmo, em seu treinamento, um conjunto de entradas e as saídas esperadas para cada caso.

2.3.3.1- *SUPPORT VECTOR MACHINES*

As Máquinas de Vetores de Suporte (*Support Vector Machines* - SVMs) consistem de uma técnica embasada na Teoria de Aprendizado Estatístico (Vapnik, 1995).

Pode-se comparar os resultados de aplicação desta técnica aos obtidos por outros algoritmos de aprendizado, como as redes neurais artificiais (Haykin, 1999).

Support Vector Machines é uma técnica que faz parte da área de Aprendizado de Máquina (AM), que é um campo de pesquisa da Inteligência Computacional que estuda o desenvolvimento de métodos capazes de extrair conceitos a partir de amostras de dados (Mitchell, 1997). Os diversos algoritmos de AM são utilizados de forma a gerar classificadores para um conjunto de exemplos. Por classificação entende-se o processo de atribuir, a uma determinada informação, o rótulo da classe – descrevendo o fenômeno de interesse, ou seja, o que se deseja aprender e fazer previsões (Baranauskas, et al. 2000) – a qual ela pertence (Russel, et al. 1995). Portanto, as técnicas de AM são empregadas na indução de um classificador, a partir de um conjunto de

treinamento, que deve ser capaz de prever a classe de instâncias quaisquer do domínio em que ele foi treinado (Lorena, et al. 2003).

Uma máquina de vetores de suporte é um conceito para um conjunto de métodos do aprendizado supervisionado que analisam os dados e reconhecem padrões. O SVM padrão prediz, para cada entrada dada, qual de duas possíveis classes a entrada faz parte, tomando como entrada um conjunto de dados. Um modelo SVM é uma representação de exemplos como pontos no espaço, mapeados de maneira que os exemplos de cada categoria sejam divididos por um espaço claro, que seja tão amplo quanto possível. Os novos exemplos são então mapeados no mesmo espaço e preditos como pertencentes a uma categoria baseados em qual o lado do espaço eles são colocados (Lorena, et al. 2003).

Em outras palavras, uma SVM encontra uma linha de separação, chamada de hiperplano, entre dados de duas classes. Conforme Figura 13, essa linha busca maximizar a distância entre os pontos mais próximos em relação a cada uma das classes (Lorena, et al. 2003).

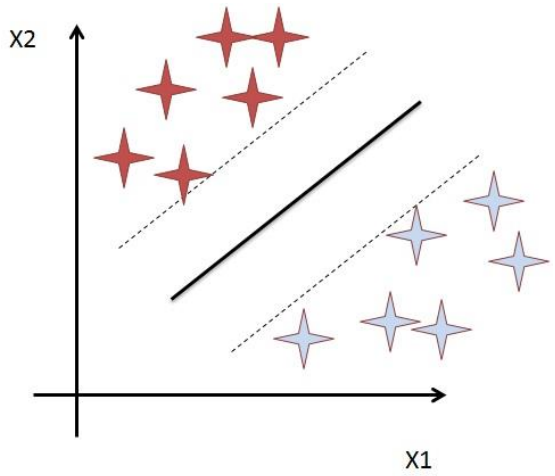

Figura 13: Hiperplano traçado entre dados de duas classes (Wikipedia: *Support Vector Machines*)

Essa distância entre o hiperplano e o primeiro ponto de cada classe costuma ser chamada de margem. A SVM primeiro classifica as classes corretamente e depois, em função dessa restrição, define a distância entre as margens. Ou seja, ela coloca em primeiro lugar a classificação das classes, definindo assim cada ponto pertencente a cada uma das classes, e em seguida maximiza a margem (Lorena, 2003).

Um conjunto de treinamento é linearmente separável se é possível separar os padrões das classes diferentes contidos no mesmo por pelo menos um

hiperplano (Russel, et al. 1995). Classificadores que separam os dados por meio de um hiperplano são denominados lineares. Também pode-se utilizar classificadores não-lineares, conforme Figura 14.

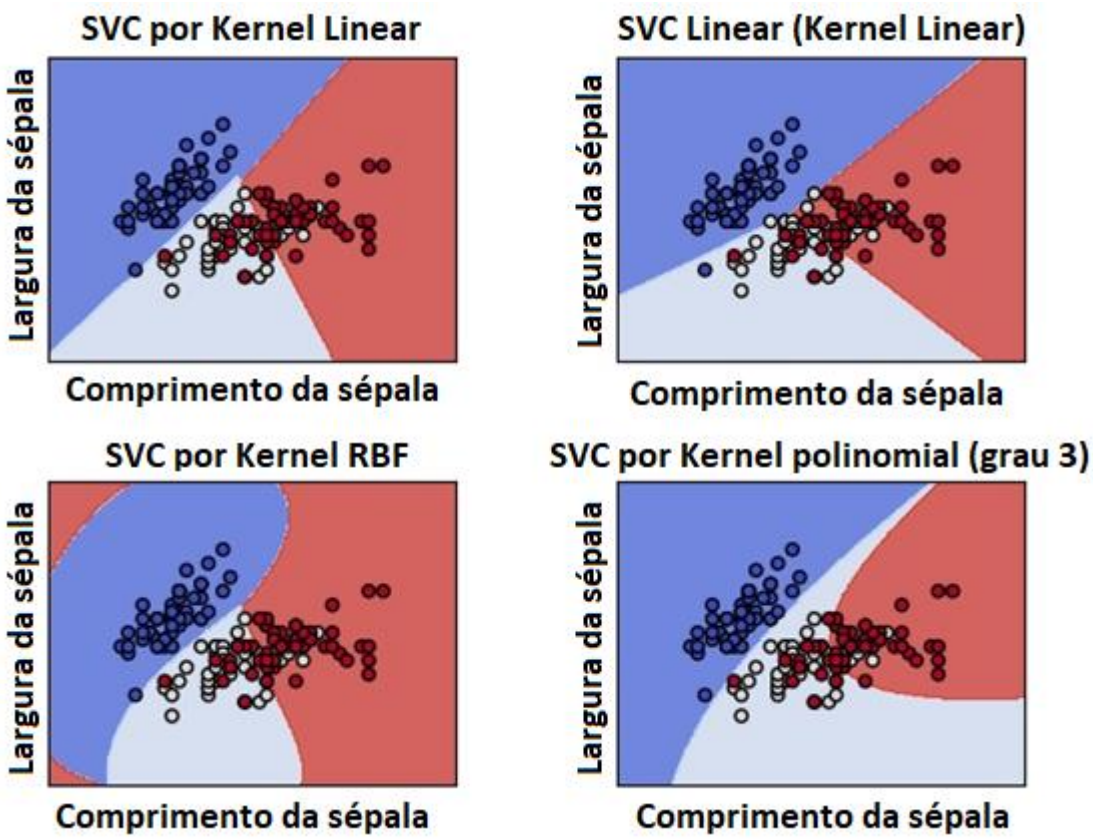

Figura 14: Exemplificação de classificadores lineares e não lineares (Scikit Learn, 2017)

É comum utilizar as funções de Kernel para o emprego de classificadores não lineares, uma vez que este tipo de classificador possui capacidade de representar espaços muito abstratos e simplicidade de cálculo (Lorena, et al. 2003). Alguns dos Kernels mais utilizados são os polinomiais, os Gaussianos ou RBF (*Radial Basis Function*) e os Sigmoidais, apresentados no Quadro 2.

Quadro 2: Funções de Kernel mais utilizadas no emprego de máquinas de vetor de suporte (Lorena, et al. 2003)

Tipo de Kernel	Função $K(\mathbf{x_i}, \mathbf{x_j})$ correspondente	Comentários
Polinomial	$(\mathbf{x_i}^T \cdot \mathbf{x_j} + 1)^p$	A potência p deve ser especificada pelo usuário
Gaussiano	$\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x_i} - \mathbf{x_j}\|^2\right)$	A amplitude σ^2 é especificada pelo usuário
Sigmoidal	$\tanh(\beta_0 \mathbf{x_i} \cdot \mathbf{x_j} + \beta_1)$	Utilizado somente para alguns valores de β_0 e β_1

Alguns pontos podem ser destacados em relação aos Kernels apresentados na Tabela 1 (Vert, 2001; Haykin, 1999; Burges, 1998):

• Em Kernels polinomiais, os mapeamentos também são funções polinomiais com complexidade crescente a medida que o expoente p aumenta.

• Por meio do emprego desse tipo de função, define-se redes neurais do tipo RBF (*Radial-Basis Function*), em que o número de funções base radiais e seus centros são determinados automaticamente pelo número de SVs (*Support Vectors*) e pelos valores de multiplicadores de Lagrange associados aos mesmos. O Kernel Gaussiano corresponde a um espaço de características de dimensão infinita. É possível afirmar que quase todas as formas de mapeamento podem ser implementadas por esta função em particular.

• A utilização do Kernel Sigmoidal, possibilita explicitar uma RNA do tipo Perceptron multicamadas. O número de neurônios da camada intermediária e os vetores de bias associados aos mesmos também são definidos pelo número de SVs e pelos valores dos multiplicadores de Lagrange associados aos mesmos, respectivamente.

2.3.3.2- REDES NEURAIS ARTIFICIAIS

Uma rede neural biológica possui vários neurônios, logo, a ideia de uma rede neural artificial é realizar a simulação da troca de sinais entre os neurônios (Barreto, 2002). A Figura 15 apresenta o exemplo de um neurônio biológico, constituído do seu corpo celular (núcleo), dendritos e axônio.

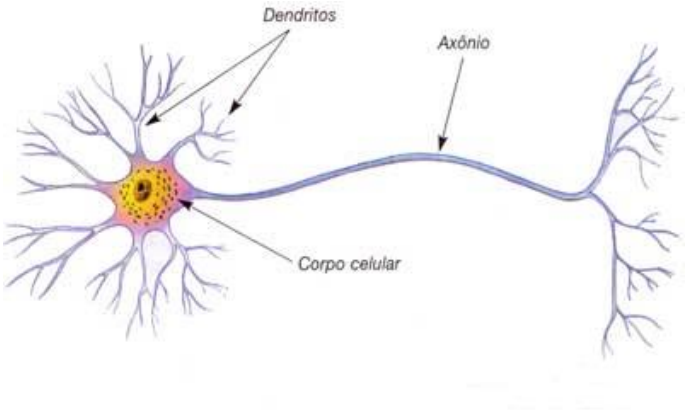

Figura 15: Estrutura celular (Granatyr, 2017)

O neurônio é a unidade utilizada para realizar o processamento de informações. Para que um neurônio se comunique com outro, o axônio efetua a transmissão de sinal (sinal elétrico, sinapses), ou seja, ele conecta os neurônios. Substâncias químicas são lançadas das sinapses e entram pelos dendritos, aumentando ou baixando o potencial elétrico do corpo da célula (Granatyr, 2017). Na Figura 16, é apresentado o fluxo de uma rede neural.

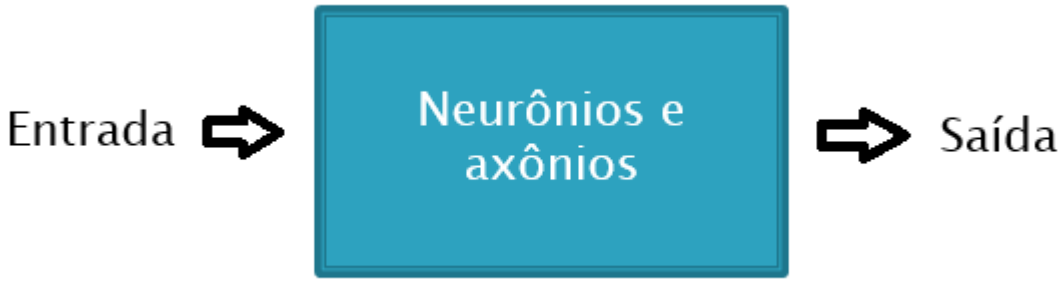

Figura 16: Parâmetros e seguimento de uma rede neural (Granatyr, 2017)

$$soma = \sum_{i=1}^{n} x_i w_i \tag{4}$$

Para a realização da simulação da aplicação das substâncias químicas nos computadores, ou seja, criar um neurônio artificial onde o potencial elétrico poderia ser controlado, foi criado o Perceptron. O Perceptron de uma camada é conhecido como o tipo mais simples de RNA *Feedforward* (é fornecido um valor de entrada, a rede processa e retorna uma resposta). Conforme Figura 17, pode-se observar sua composição. As entradas são os parâmetros a serem classificados; os pesos representam o sinal a ser transmitido, ou seja, as sinapses; a função soma é representada pela eq. (4); e a função de ativação que define se o neurônio será ativado quando a sua entrada é maior que um número pré-definido.

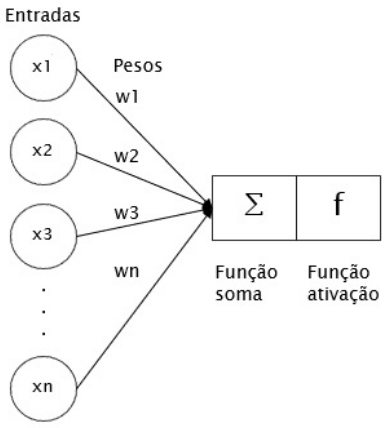

Figura 17: Perceptron de uma camada (Granatyr, 2017)

$$erro = s_d - s_c \quad (5)$$

$$p_{(n+1)} = p_n + (x \times ent \times erro) \quad (6)$$

Em que:

O $erro$ é a diferença entre a saída esperada e a saída calculada;

s_d a saída desejada;

s_c a saída calculada;

$p_{(n+1)}$ é o valor do peso atualizado;

p_n é o valor do peso atual;

x é um valor definido pelo usuário, chamado de taxa de aprendizagem;

ent é o valor de entrada do atual peso que está sendo atualizado.

A RNA trata de aprender um melhor conjunto de pesos para uma determinada base de dados, ou seja, todo o conhecimento adquirido na fase de treinamento é armazenado através da atualização dos valores dos pesos, buscando encontrar uma melhor combinação de valores relacionando as entradas com as saídas (Ferneda, 2006). Para atualização do valor de cada peso, conforme eq. 6, deve-se primeiro calcular o valor do erro, de acordo com eq. 5.

Enquanto o erro for diferente de zero, a atualização dos pesos é realizada. O Algoritmo 1 resume o funcionamento de um Perceptron de uma camada.

Algoritmo 1: Funcionamento de um Perceptron de uma camada

$Enquanto\ o\ erro\ for\ diferente\ de\ zero$:

$Para\ cada\ registro$:

$Calcula\ a\ saída\ com\ os\ pesos\ atuais$

$Compara\ a\ saída\ calculada\ com\ a\ saída\ desejada, somando\ o\ erro$

$Para\ cada\ peso\ da\ rede$:

$Atualiza\ o\ peso$: $p_{(n+1)} = p_n + (x \times ent \times erro)$

Porém, este tipo de algoritmo não funciona na maioria dos casos. O Perceptron de uma camada é um algoritmo que é eficaz apenas em casos linearmente separáveis. Como acontece na Figura 12, por exemplo, este algoritmo só funciona em casos em que as classes podem ser totalmente separadas por uma linha. Na Figura 13 é possível observar que algumas classes não foram separadas adequadamente (algumas brancas entre as vermelhas), ou seja, este algoritmo não seria eficaz para este tipo de problema.

Como solução, foram propostas as redes neurais multicamada. Uma rede neural multicamada, além da camada de entrada e de saída, possui camadas ocultas. Uma rede neural artificial é chamada de multicamada quando possui pelo menos uma camada oculta (Fleck, et al. 2016). A Figura 18 apresenta o exemplo de uma RNA com uma camada oculta.

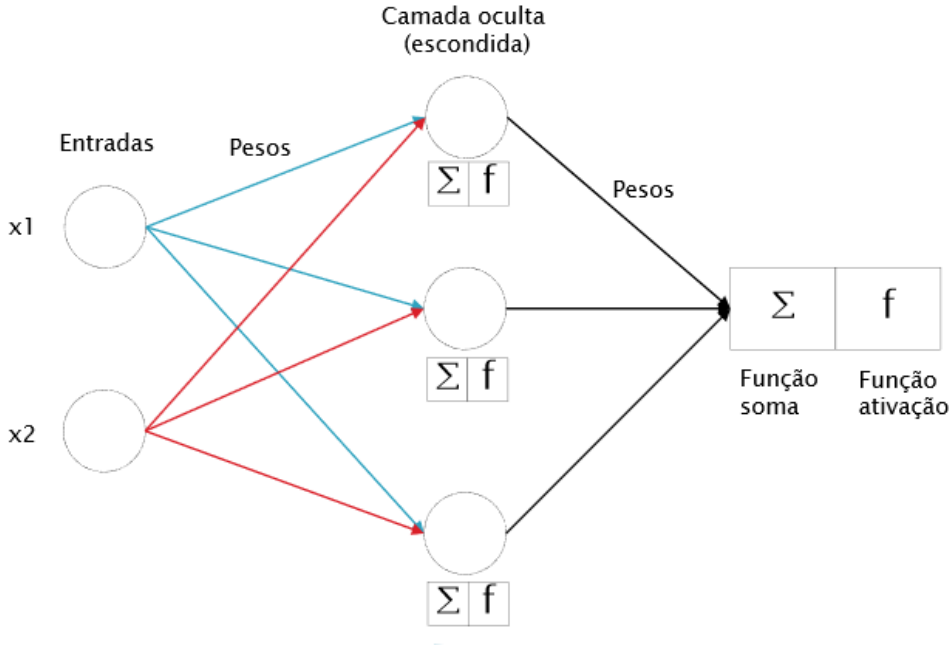

Figura 18: Rede neural artificial multicamada com uma camada oculta (Granatyr, 2017)

Observando a figura, é possível notar que cada um dos neurônios da camada oculta trabalha como a camada de saída de uma rede de uma camada, onde todos realizam as somatórias do produto dos pesos com as entradas e executam a função de ativação. A diferença é que, em seguida, é realizada a somatória do produto de cada uma das respostas da função de ativação com os pesos seguintes e é aplicada mais uma função de ativação. Se a rede possuísse outras camadas ocultas, seria realizado o mesmo processo, em que cada um dos neurônios de cada camada oculta realizaria a função soma e a função de ativação.

Existem vários tipos de funções de ativação. Algumas das mais conhecidas são: step, sigmoide e tangente hiperbólica (Fleck, et al. 2016). A Figura 19 apresenta a resposta de cada uma destas funções.

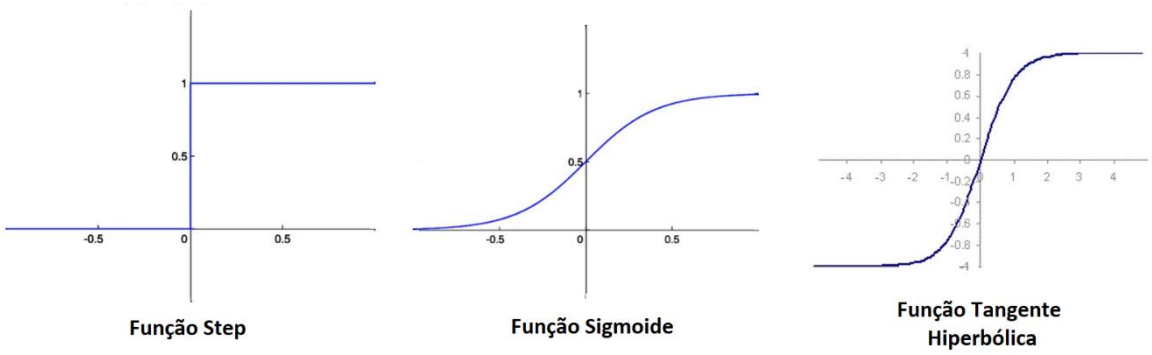

Figura 19: Exemplos de função de ativação (Granatyr, 2017)

Na função step (degrau), para valores maiores que zero, é retornado o valor 1 e, em caso contrário, é retornado o valor zero. Na função sigmoide é aplicada a eq. (7) onde, se x for alto, a resposta será aproximadamente 1 e, se for baixo, aproximadamente zero, não retornando valores negativos. Já na função tangente hiperbólica é aplicada a eq. (8), onde os valores de resposta se situam entre -1 e 1.

$$y = \frac{1}{1 + e^{-x}} \tag{7}$$

$$y = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{8}$$

Em seguida, após o cálculo da resposta pela função de ativação, é calculado o valor do erro pela eq. (5) para atualização de todos os pesos pelo processo de *backpropagation*, repetindo todo o processo. A implementação só é finalizada quando o valor de erro for mínimo.

Para simplificar o processo, é apresentado o Algoritmo 2 e o fluxograma da Figura 20, que trata de uma rede neural *feedforward* multicamada.

Algoritmo 2: Funcionamento de uma rede neural *feedforward* multicamada

$Inicializa\ os\ pesos\ com\ valores\ aleatórios$

$Baseado\ nos\ dados, realiza\ os\ cálculos\ com\ os\ pesos$

$Calcula\ o\ erro$

$Calcula\ a\ mudança\ nos\ pesos\ e\ os\ atualiza\ (backpropagation)$

$O\ algoritmo\ termina\ quando\ o\ erro$ é $mínimo$

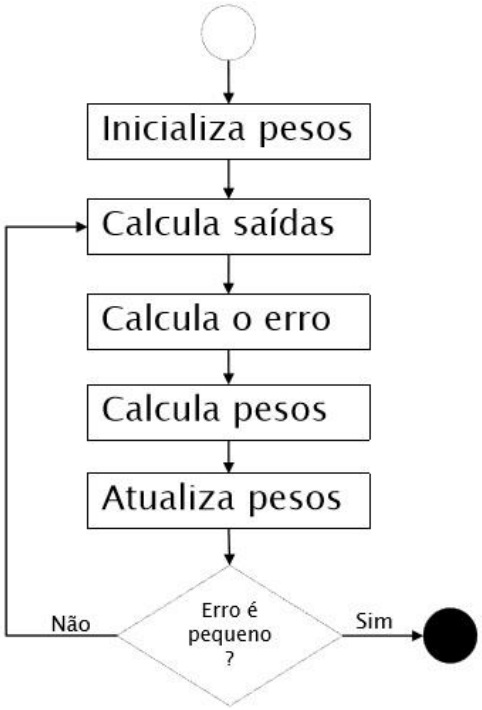

Figura 20: Fluxograma do funcionamento de uma rede neural *feedforward* multicamada (Granatyr, 2017)

Uma forma de garantir que o erro mínimo será encontrado é aplicando o conceito de descida do gradiente (*gradient descent*). Para exemplificar, primeiro, é estabelecido na Figura 21 que o ponto 1 é onde está situado o erro inicial, onde os valores de pesos são inicializados. Ao se atualizar os valores de peso, é encontrado um novo erro, situado no ponto 2. O objetivo é realizar esta atualização de pesos de tal forma que seja encontrado os valores que resultem no erro situado no ponto 3, que representa o erro mínimo.

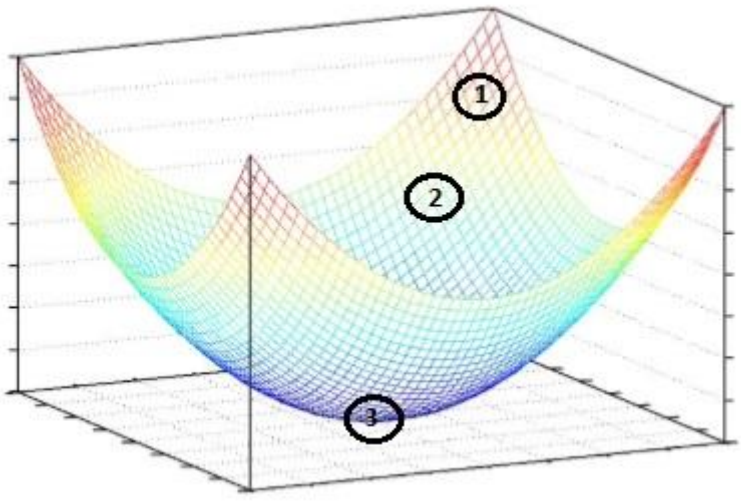

Figura 21: Exemplificação da descida do gradiente (Granatyr, 2017)

A figura 22 apresenta outro exemplo de aplicação de descida do gradiente, onde os pesos são atualizados, de forma a “descer” buscando o valor de erro mínimo.

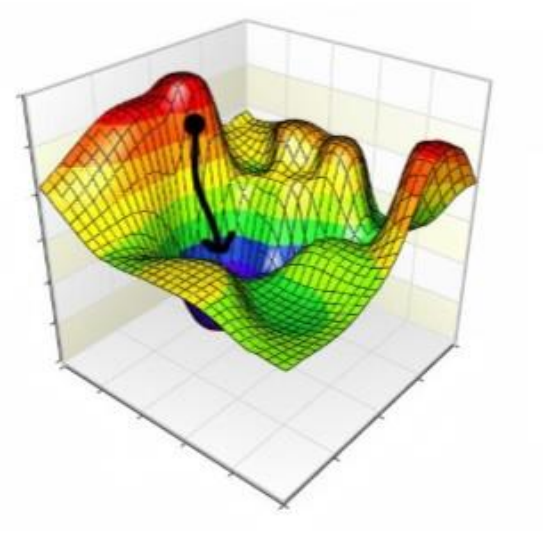

Figura 22: Exemplificação da descida do gradiente (Granatyr, 2017)

O objetivo do cálculo do gradiente é atualizar os pesos de forma a encontrar um erro mínimo. Este cálculo é realizado para definir quanto ajustar os pesos. Porém, conforme pode ser observado na Figura 23, existem casos que possuem pontos chamados de “mínimos locais”. Estes pontos, que aparentam ser o mais baixo, podem “enganar” o algoritmo. O local com erro mínimo buscado é chamado de “mínimo global”. Ao se iniciar a aplicação com um valor de erro (como o mostrado em verde, na figura), o algoritmo deve se cautelar em relação à direção que irá tomar ao iniciar a descida, pois pode resultar em um “mínimo local”. Para resolver este problema, é realizado o cálculo do declive da curva aplicando derivadas parciais.

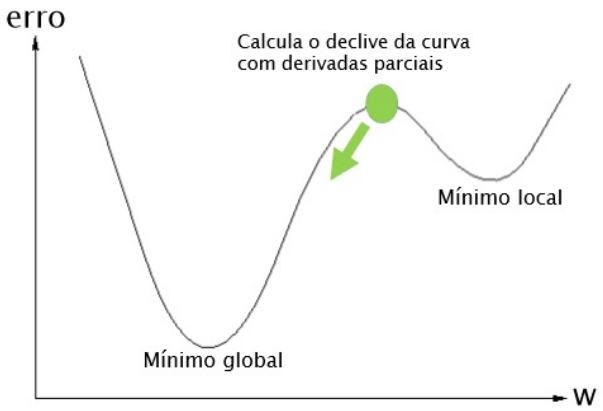

Figura 23: Exemplificação da diferença entre mínimo local e mínimo global (Granatyr, 2017)

O cálculo da derivada do gradiente é realizado justamente para direcionar a descida para o lado correto. Porém, este não é o único passo para realização da descida do gradiente. Seguindo o fluxograma da Figura 24, é possível encontrar o valor de gradiente.

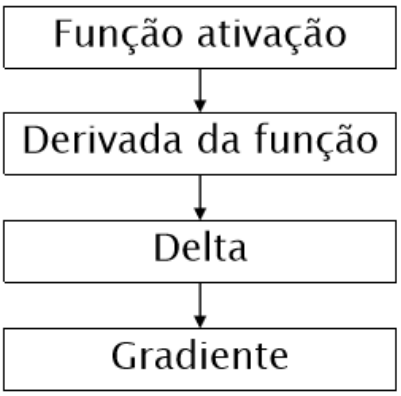

Figura 24: Fluxograma para realização da descida do gradiente (Granatyr, 2017)

$$\Delta_s = erro \times d \tag{9}$$

$$\Delta_o = d \times peso \times \Delta_s \tag{10}$$

O próximo passo é realizar o cálculo do valor Delta (Δ). Este parâmetro deve ser calculado tanto na camada de saída (Δ_s), conforme eq. (9), quanto na camada oculta (Δ_o) da RNA, conforme eq. (10). Lembrando que d é o valor calculado através da derivada parcial da função de ativação aplicada.

Alguns conceitos já citados anteriormente, mas que não foram vistos com muito detalhe são *feedforward* e *backpropagation*. Conforme Figura 25, pode-se dizer que o processo *feedforward* são todos os cálculos que ocorrem da esquerda para a direita, ou seja, as aplicações das funções de soma e ativação nas camadas ocultas e na camada de saída, assim como o cálculo do erro. Já o processo *backpropagation* incorpora todos os cálculos realizados da direita para a esquerda, tal como a atualização de todos os pesos da RNA.

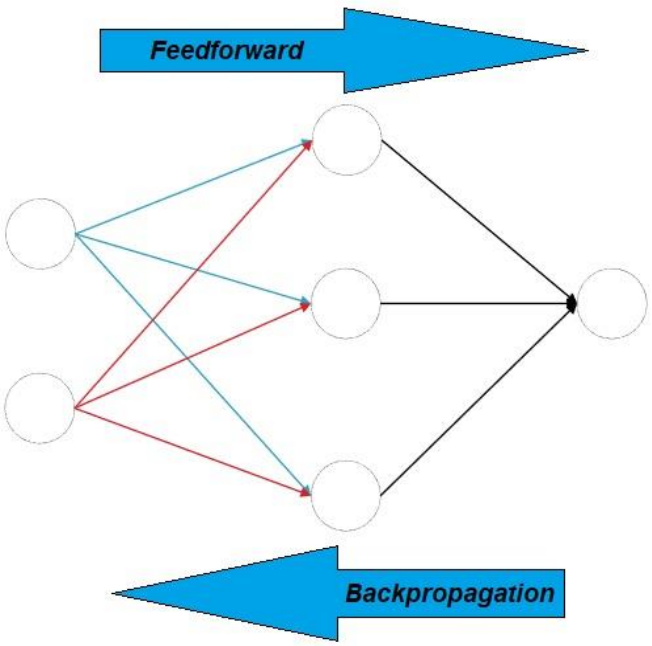

Figura 25: Relação *feedforward* e *backpropagation* em uma rede neural artificial (próprio autor)

$$p_{n+1} = (p_n \times m) + (ent \times \Delta \times x) \tag{11}$$

Para o processo de *backpropagation* deve-se definir valores para os parâmetros taxa de aprendizagem (*learning rate*) e momento (*momentum*), uma vez que são necessários à eq. (11) para atualização dos pesos (Granatyr, 2017):

• x é a taxa de aprendizagem, responsável por definir quão rápido o algoritmo vai aprender. Se o valor definido for alto, a convergência será rápida, mas pode perder o mínimo global. Se for baixo, será mais lento, mas tem mais chances de chegar no mínimo global.

• m é o momento, responsável por escapar de mínimos locais, porém, nem sempre funciona, além de definir o quão confiável é a última alteração. Se o valor dado a este parâmetro for alto, a velocidade de convergência aumenta. Se for baixo, pode evitar mínimos locais.

• Δ deve ser o calculado na camada em que os pesos estão sendo atualizados. Para atualização dos pesos entre a camada de saída e a camada oculta, utiliza-se o delta de saída, conforme eq. (9). Para a atualização dos pesos entre a camada oculta e a camada de entrada, utiliza-se o delta da camada oculta, conforme eq. (10).

• O parâmetro ent de entrada, ao se realizar a atualização dos pesos entre a camada de saída e a camada oculta, se refere ao valor de resposta da função de ativação do neurônio da camada oculta em que o peso está sendo atualizado. Ao se realizar a atualização dos pesos entre a camada oculta e a camada de entrada, se refere ao valor de entrada relacionado ao peso que está sendo atualizado.

• $peso_{(n+1)}$ se refere ao valor do peso após atualização e $peso_n$ é o valor do peso antes da atualização.

Um outro conceito muito utilizado em algoritmos de redes neurais artificiais é a implementação de *bias*, que é o ato de acrescentar uma nova entrada adicional em cada uma das camadas da rede. Empregando o *bias*, aumenta-se os graus de liberdade, permitindo uma melhor adaptação, por parte da RNA, ao conhecimento a ela fornecido.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2 \tag{12}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2} \tag{13}$$

Para o cálculo do erro não se é restringido a utilizar apenas a eq. (5). Pode-se utilizar MSE (*Mean Square Error*), conforme eq. (12) ou RMSE (*Root Mean Square Error*), conforme eq. (13), por exemplo (Granatyr, 2017). Desta forma, N representa o número total de amostras, f_i a saída esperada e y_i a saída calcula.

Deve-se ressaltar também que alguns problemas irão necessitar de mais de um neurônio na camada de saída. Porém, quando se tem mais de um neurônio na camada de saída, é necessário realizar uma codificação para interpretar o resultado. Por exemplo, se uma RNA tem a função de diferenciar imagens contendo as letras A, B e C, tem-se mais de uma resposta na saída, ou seja, ela não é binária, são três opções de resposta. Para este tipo de problema pode-se empregar três neurônios na camada de saída e, ao se obter a resposta (1,0,0), sendo cada algarismo a resposta de um neurônio, entende-se como a letra A. Para a letra B espera-se obter a resposta (0,1,0) e para a C (0,0,1).

Da mesma forma, existem problemas que necessitarão de mais do que apenas uma camada oculta. Redes neurais com mais de uma camada oculta, também conhecidas como *deep learnings,* exigem técnicas um pouco diferente das já tratadas aqui. A Figura 26 exemplifica uma dessas redes. Lembrando que,

problemas linearmente separáveis não necessitam de camadas ocultas. Quanto mais complexo o problema, mais camadas ocultas é recomendado de se utilizar (Fleck, et al. 2016).

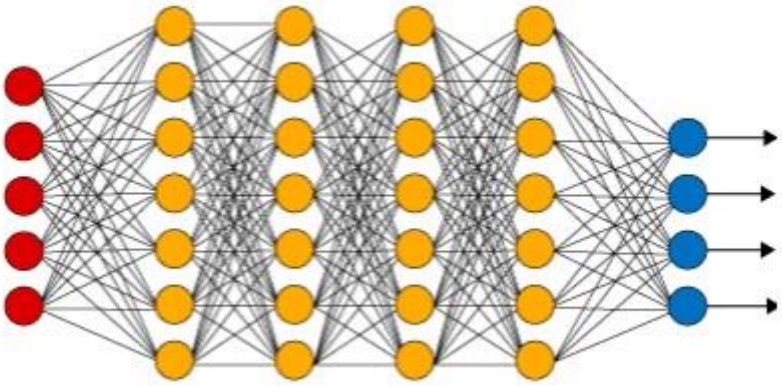

Figura 26: Rede neural multicamada com quatro camadas ocultas e quatro neurônios na camada de saída (Granatyr, 2017)

3- DESENVOLVIMENTO

Como citado no capítulo referente à Metodologia, para esta primeira etapa do projeto foi considerada a análise e comparação da eficiência de códigos empregando o método SVM e de redes neurais multicamada para um mesmo conjunto de objetos.

Primeiramente, foi utilizado o banco de dados MNIST, que é um conjunto de imagens de algarismos de 0 a 9 escritos a mão, conforme pode ser observado na Figura 27. Este é um banco de dados extenso, constituído de 42000 amostras, aplicado por iniciantes que estão adentrando os estudos em técnicas de classificação de imagens (LeCun, et al. 1998).

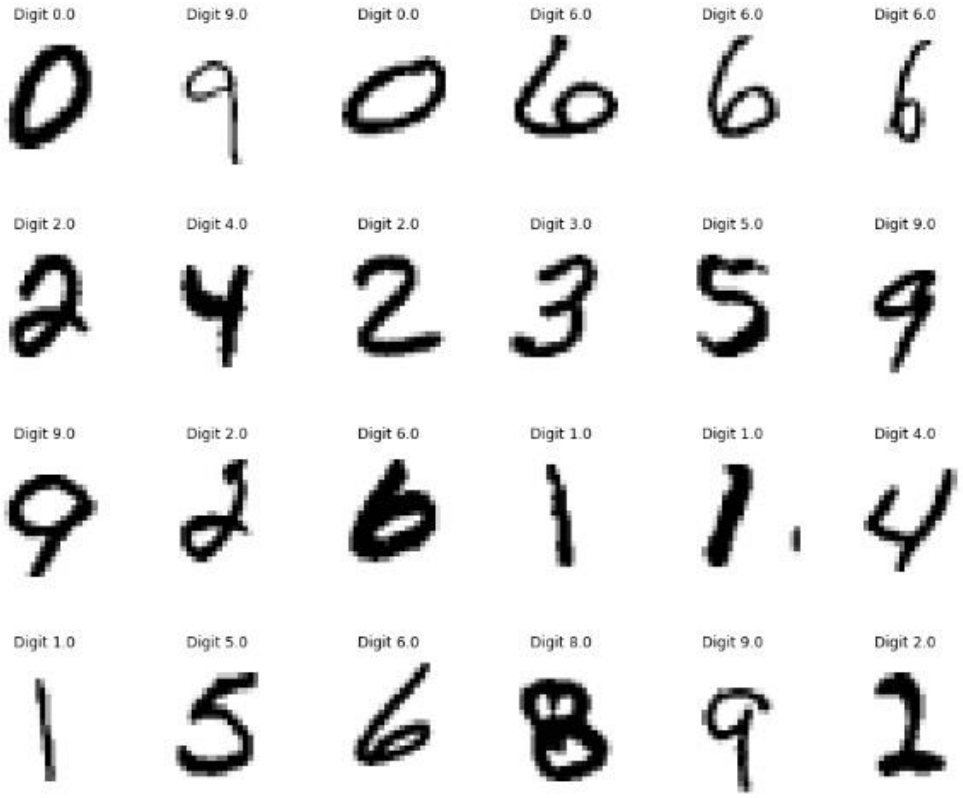

Figura 27: Exemplificação do banco de dados MNIST (LeCun, et al. 1998)

MNIST é um conjunto de dados que vem sido utilizado popularmente como referência de comparação de eficiência entre algoritmos que são desenvolvidos ao longo dos anos. O Quadro 3 apresenta uma relação entre diversos tipos de redes neurais artificiais e máquinas de vetores de suporte e a relação de acurácia entre eles, ao serem aplicados no MNIST.

Quadro 3: Relação de taxa de erro para algoritmos aplicados ao MNIST

CLASSIFICADOR	PRÉ-PROCESSAMENTO	TAXA DE ERRO NOS TESTES (%)	Referência
Classificador Linear			
Classificador Linear (Rede Neural de 1 camada)	Nenhum	12	LeCun et al. 1998
Classificador Linear (Rede Neural de 1 camada)	Correção inclinação (deskewing)	8,4	LeCun et al. 1998
SVMs			
SVM, Kernel Gaussiano	Nenhum	1,4	LeCun et al. 1998
SVM, Polinomial (grau 4)	Correção inclinação (deskewing)	1,1	LeCun et al. 1998
Redes Neurais Artificiais			
2 camadas, 300 neurônios na camada oculta	Nenhum	4,7	LeCun et al. 1998
2 camadas, 1000 neurônios na camada oculta	Nenhum	1,5	LeCun et al. 1998
3 camadas, 300+100 neurônios nas camadas ocultas	Nenhum	3,05	LeCun et al. 1998

Como pode ser observado no quadro apresentado, o Perceptron de uma camada (*1-layer NN*), sem aplicação de pré-processamento na imagem, apresentou acurácia de 88%, enquanto que, com aplicação de *deskewing* (correção de inclinação na imagem), resultou em 91,6%. Nos testes em questão, ao invés de 42000 amostras, foram empregadas 70000 amostras, sendo 60000 para a etapa de treinamento do algoritmo e 10000 para testes. Ao dizer que foi obtida acurácia de 91,6%, indica-se que foram realizados 9160 acertos dentro das 10000 tentativas.

Na aplicação de SVMs, foram utilizados os métodos RBF (*Gaussian Kernel*) e polinomial, já citados anteriormente, conforme visto na Figura 13. Para o método RBF, foi alcançada acurácia de 98,6%, e para o método polinomial com *deskewing*, 98,9%.

Por fim, empregando RNAs multicamada, com uma camada oculta (*2-layer NN*, referindo-se à camada oculta e à camada de saída) possuindo 300 neurônios, o resultado foi de 95,3% de acurácia. Com a camada oculta possuindo 1000 neurônios, a resposta foi de 95,5%. Por fim, para uma RNA com duas camada ocultas, com a primeira possuindo 300 neurônios e a segunda 100, foi encontrado o valor de 96,95%.

Nos testes realizados neste trabalho, foi utilizado um banco de dados do MNIST menor (42000 amostras, divididas entre treinamento e teste), devido a isto, esperava-se que a acurácia resultasse em um valor razoavelmente menor. Nestes testes, foram utilizados algoritmos de SVM pelo método RBF, uma vez que, para comparação com o apresentado no Quadro 3, não foi utilizado pré-processamento. Em relação às RNAs, foi empregada uma rede multicamada com uma camada oculta possuindo 100 neurônios.

Além do banco de dados da MNIST, também foram empregados testes em imagens de objetos do cotidiano. Para estas imagens foi realizado um pré-processamento simples:

• Conversão de RGB para *Grayscale*.

• Redução da resolução de 1090x1080 pixels para 192x108, 96x54, 48x27 e, em alguns casos, para 480x270.

• Normalização dos valores dos pixels de 0 a 255 para 0 a 1.

No caso do algoritmo das RNAs, deve-se estabelecer um valor para a variável “época”. Nestes testes foi estabelecido o valor 10. Esta variável é responsável por determinar quantas vezes os pesos serão atualizados, ou seja, o número de iterações para cálculo das funções de soma, funções de ativação, erro, delta, gradiente e novos pesos é dado como uma época.

Para comparação dos resultados, os únicos parâmetros a serem modificados em cada teste são: o número de amostras para treinamento e testes, a resolução da imagem e a taxa de aprendizagem. Os outros parâmetros foram mantidos constantes para uma melhor análise comparativa.

Após os testes com MNIST, foram realizados testes com uma caneta. Para a classificação, foram empregadas imagens da caneta com e sem tampa, julgando que a ausência da tampa poderia ser categorizada como um tipo de deformidade. Na Figura 28 são apresentados exemplos das imagens utilizadas. Foi realizado o rotacionamento do objeto para evitar que o algoritmo entrasse em *overfitting*, sendo capaz de realizar a identificação em diversas posições.

Figura 28: Exemplos de imagens empregadas para classificação da caneta (próprio autor)

Em seguida, utilizando a embalagem de um jogo digital como objeto de estudo, foi realizada a classificação do objeto com e sem rótulo, considerando a ausência do rótulo como um tipo de deformidade. A Figura 29 apresenta alguns exemplos de imagens de amostragem aplicados ao algoritmo.

Figura 29: Exemplos de imagens empregadas para classificação da embalagem do jogo ditigal (próprio autor)

Utilizando a mesma embalagem, foi realizada a classificação das imagens levando em consideração seu interior. Como dentro do recipiente é apresentado

um manual e o cartucho do jogo, conforme Figura 30, sua categorização foi disposta em 4 casos diferentes:

- Com jogo e manual (1).
- Sem jogo e com manual (2).
- Com jogo e sem manual (3).
- Sem jogo e manual (4).

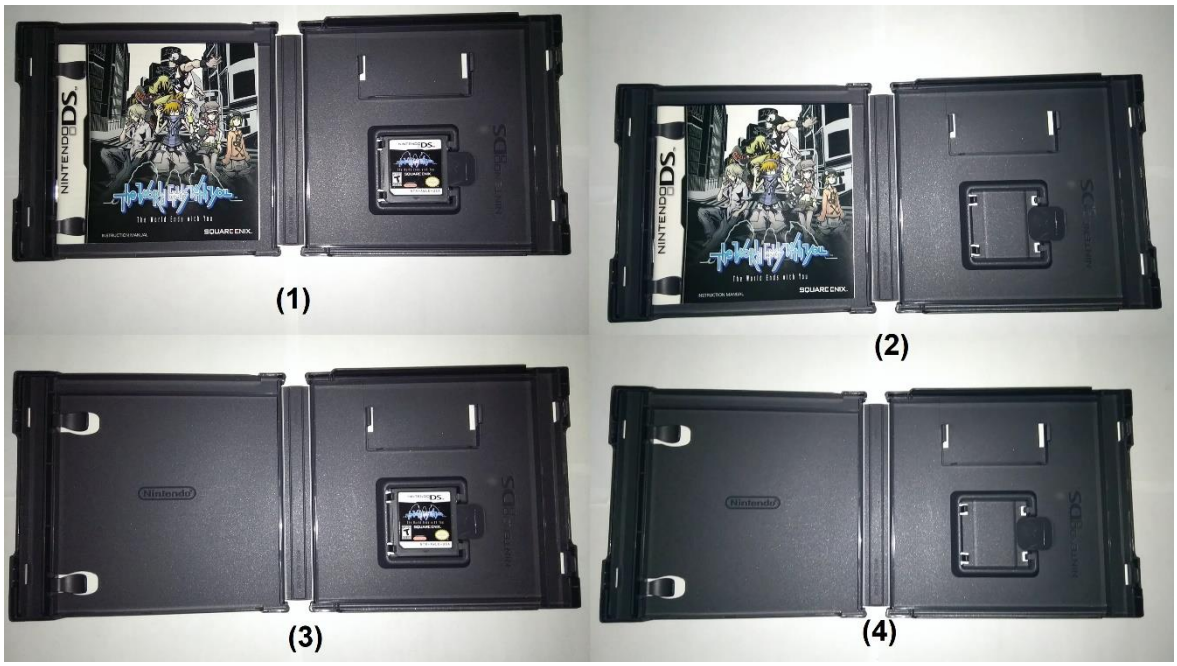

Figura 30: Exemplos de imagens empregadas para classificação do interior da embalagem do jogo ditigal (próprio autor)

Ainda tratando do interior da embalagem, foram realizados mais testes categorizando as classes de outras duas formas diferentes. A primeira, considerando a embalagem preenchida para os casos (1), (2) e (3) e não preenchida para o caso (4). A segunda, descartando totalmente todas as amostras dos casos (2) e (3) e realizando o treinamento e teste apenas com as amostras dos casos (1) e (4), julgando a embalagem como estando cheia ou vazia (resposta binária).

O teste seguinte foi realizado com 4 bonecos diferentes, conforme Figura 31. A classificação foi realizada para quatro saídas, ou seja, o algoritmo deveria aprender como identificar cada um dos objetos.

Figura 31: Exemplos de imagens empregadas para classificação dos bonecos (próprio autor)

Por fim, o último teste realizado foi utilizando o boneco da Figura 32. Conforme pode ser observado, foi realizada a classificação de acordo com a ausência ou não de seus braços.

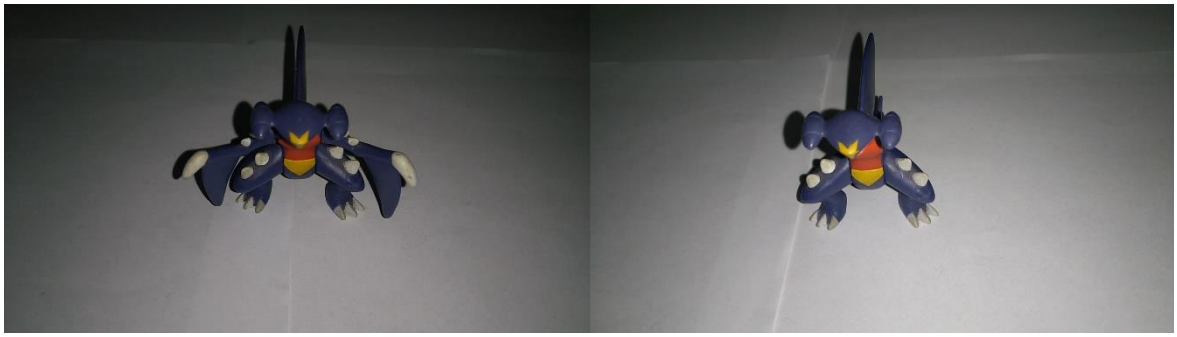

Figura 32: Exemplos de imagens empregadas para classificação do boneco (próprio autor)

O Algoritmo 3 foi aplicado para ambos os testes (SVM e redes neurais multicamada), com exceção da linha de definição do valor de épocas para o SVM. Para obtenção das imagens aplicadas como banco de dados (caneta, embalagem do jogo, bonecos) foi realizada a gravação de um vídeo do objeto e a separação de seus *frames*.

Algoritmo 3: Cálculo da acurácia da classificação de imagens

Separação dos frames dos vídeos para obtenção das imagens
Conversão das imagens de RGB para Grayscale
Redimensionamento das imagens para a resolução desejada
Normalização dos pixels para valores de 0 *a* 1
Definição dos valores de época e taxa de aprendizagem
Enquanto o número de épocas é menor que o definido:
    Emprego da rotina de treinamento pelo método desejado
Emprego da rotina de teste
Para todos os testes realizados:
    Definir acurácia total

Para a segunda etapa, observou-se a necessidade de correção de alguns fatores para melhor interação entre a plataforma e o usuário, além de alterações no processamento e tratamento dos dados.

Primeiramente, em relação à interação com o usuário, optou-se pelo desenvolvimento de uma interface gráfica. Para tal, foi selecionado o *Tkinter*, que nada mais é do que uma biblioteca da linguagem *Python* específica para desenvolvimento de GUIs (*Graphic User Interface* – Interface Gráfica com o usuário) (Ferg, 2009). Esta aplicação foi necessária para auxiliar ao usário na definição de parâmetros, uma vez que a alteração direta dos parâmetros em linha de código pode ser confusa e precisar de certo conhecimento básico de programação. A Figura 33 apresenta a janela inicial da Interface, enquanto que a Figura 34 apresenta as janelas seguintes ao definir o algoritmo de classificação a ser empregado.

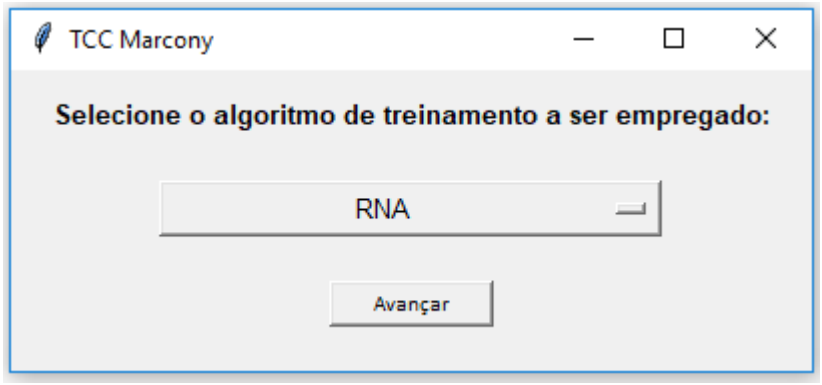

Figura 33: Janela inicial para seleção do algoritmo de classificação (próprio autor)

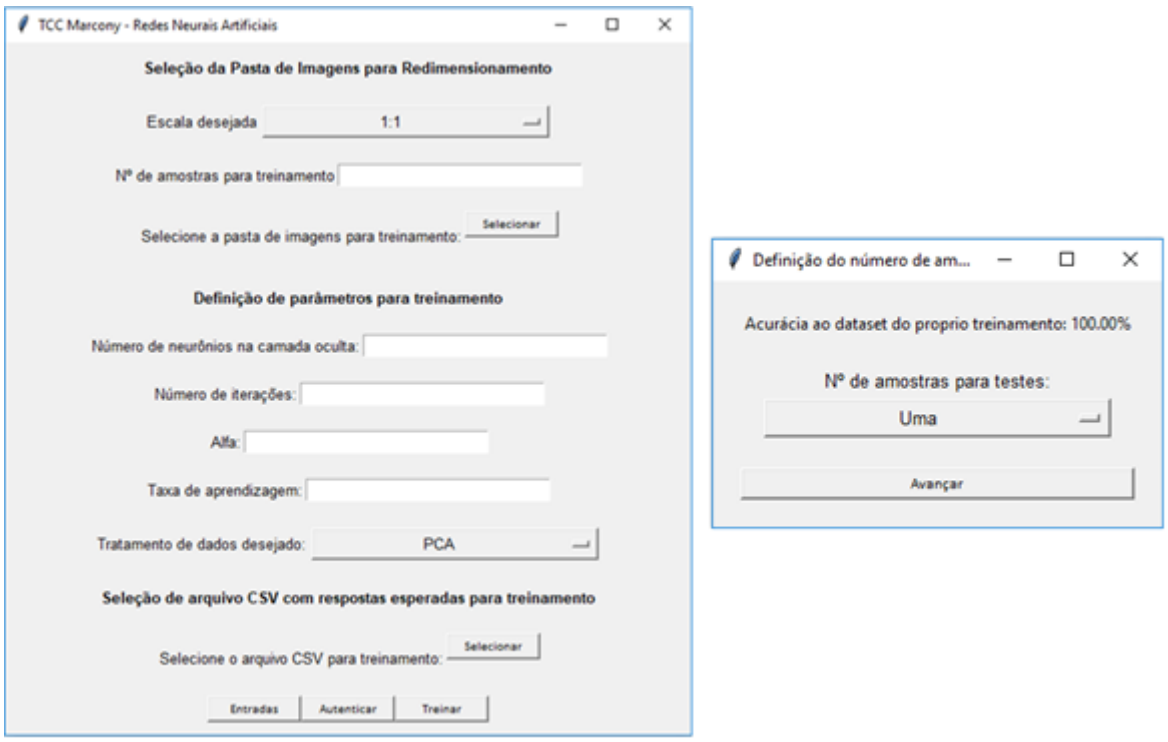

Figura 34: Janela para entrada de parâmetros para treinamento por RNA (próprio autor)

Ao definir os parâmetros adequadamente, autenticá-los e selecionar a opção “Treinar”, o treinamento é realizado. É exibido o valor de acurácia do algoritmo em relação aos dados do próprio treinamento e são apresentadas as opções de teste para uma imagem específica (retornando sua previsão) ou para várias (que calcula a acurácia do algoritmo para um conjunto de respostas pré-definidas). O arquivo CSV que correponde às respostas para treinamento trata de uma matriz coluna, onde cada linha corresponde à resposta esperada de uma figura.

Em relação ao processamento e tratamento de dados, notou-se que a quantidade exuberante de variáveis de entrada (20736, para imagens em escala 192x108 pixels e 129600, para 480x270 pixels, por exemplo) era um fator decisivo, tanto para o desempenho do código em questão de tempo de execução, quanto para questões de acurácia. Um número maior de entradas exige uma quantidade elevada de neurônios e, consequentemente, mais pesos a serem a atualizados a cada iteração. Da mesma forma, existe chances de, quanto maior o número de entradas normalizadas, menor ser a variância entre elas, dificultando a previsão correta do algoritmo.

$$r = \frac{\sum_{i=1}^{N} R_i}{\sum_{i=1}^{N} R_i + \sum_{i=1}^{N} G_i + \sum_{i=1}^{N} B_i} \tag{14}$$

$$R_m = \frac{\left(\frac{\sum_{i=1}^{N} R_i}{N}\right)}{255} \quad (15)$$

Para a redução significativa do número de variáveis de entrada, foi aplicado o tratamento apresentado no fluxograma da Figura 35. As coordenadas cromáticas foram calculadas através da equação (14) e os pixels médios através da equação (15) (Louro, et al. 2006), onde, neste caso, estão sendo calculados os valores para o canal vermelho, representado pela letra R. Para a coordenada cromática referente ao canal vermelho, é realizada a somatória de todos os pixels e o resultado é dividido pela soma das somatórias de cada canal. Para os pixels médios, é realizada a somatória de todos os pixels do canal, o resultado é dividido pelo número total N de pixels e normalizado, sendo dividido por 255.

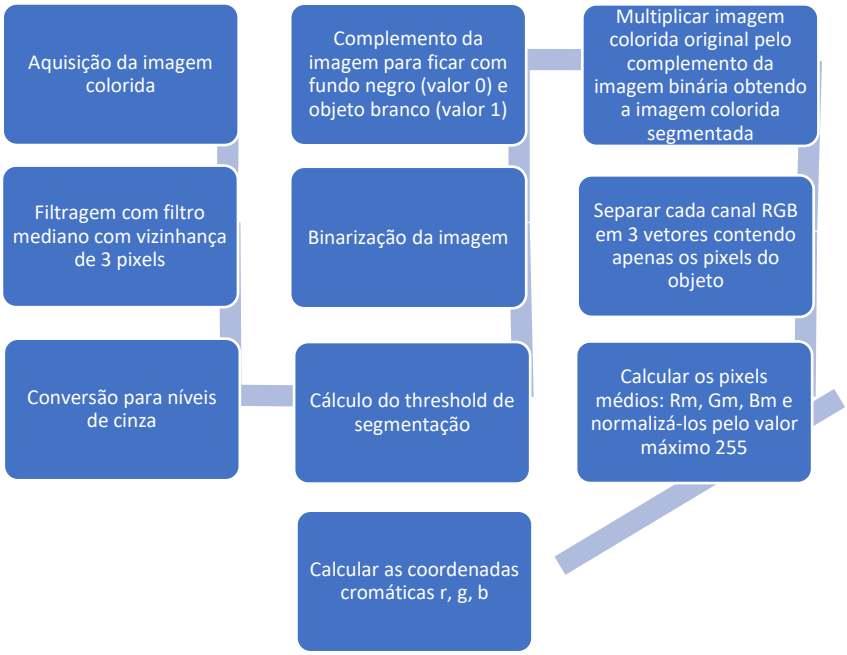

Figura 35: Fluxograma do tratamento de imagem para extração de caraterísticas (Louro, et al. 2006, alterado pelo autor)

Desta forma, foram utilizadas como variáveis de entrada do algoritmo de classificação os pixels médios e as coordenadas cromáticas, reduzindo o número de entradas para apenas 6. A Figura 36 apresenta um exemplo de tratamento de uma das imagens abordadas para classificação do refil de canetas para aplicação de insulina, aplicando o método citado.

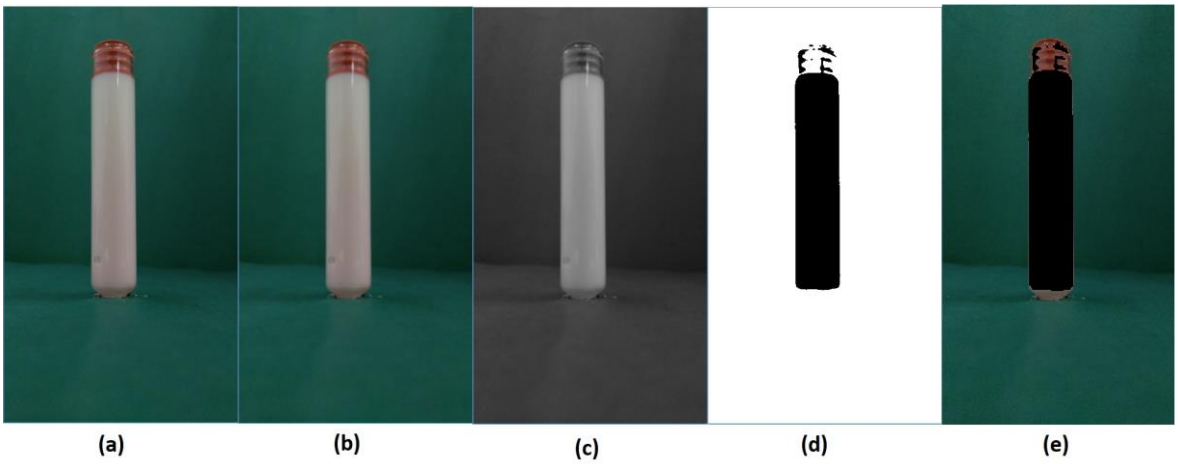

Figura 36: Exemplo de tratamento de imagem, representando: (a) a imagem original colorida; (b) imagem após filtragem com filtro mediano com vizinhança de 3 pixels; (c) imagem convertida para níveis de cinza; (d) imagem binarizada; (e) imagem colorida segmentada (próprio autor)

Para o caso dos refis de canetas de aplicação de insulina, foram observadas duas características para classificação: a coloração do fluido contido em seu interior e o seu volume. Através da coloração é possível notar se o produto está vencido, uma vez que a coloração normal para a insulina regular é transparente e, qualquer coisa diferente disso, não é aceitável. Se o fluido apresentar cor turva ou esbranquiçada, está vencido (Humulin®, 2018), e, se apresentar qualquer outra variação de cor, especula-se que pode ter ocorrido algum erro nos processos químicos. Em relação à quantidade, não é desejável que o produto, ao ser comprado, apresente menor volume do que o especificado em sua embalagem.

Visando classificar o produto através destas duas características, considerando o tratamento apresentado na Figura 36, foram aplicadas duas redes neurais sequencias: a primeira responsável por tratar da verificação da coloração e a segunda da análise da altura. A primeira rede neural apresentando 6 entradas, uma camada oculta de número de neurônios definido pelo usuário e uma camada de saída com um neurônio. Conforme Figura 37, a saída da primeira classificação é dada como entrada na segunda rede neural, que possui também como entrada o valor de altura do fluido. Para verificação da altura foi estipulada uma área de referência na imagem onde espera-se que a borracha utilizada como tampa esteja quando o refil está cheio, já que a posição do refil é padronizada. Quando o volume de fluido diminui, a borracha acompanha o movimento de descida, logo, quando o fluido estiver abaixo do nível considerado como ideal, a

borracha não estará dentro da área estipulada. Assim, é feito o cálculo da média dos pixels dentro desta área para verificação da altura e o resultado é dado como entrada na segunda rede neural. Da mesma forma, para o SVM, também foram aplicados dois treinamentos sequenciais e classificações sequenciais.

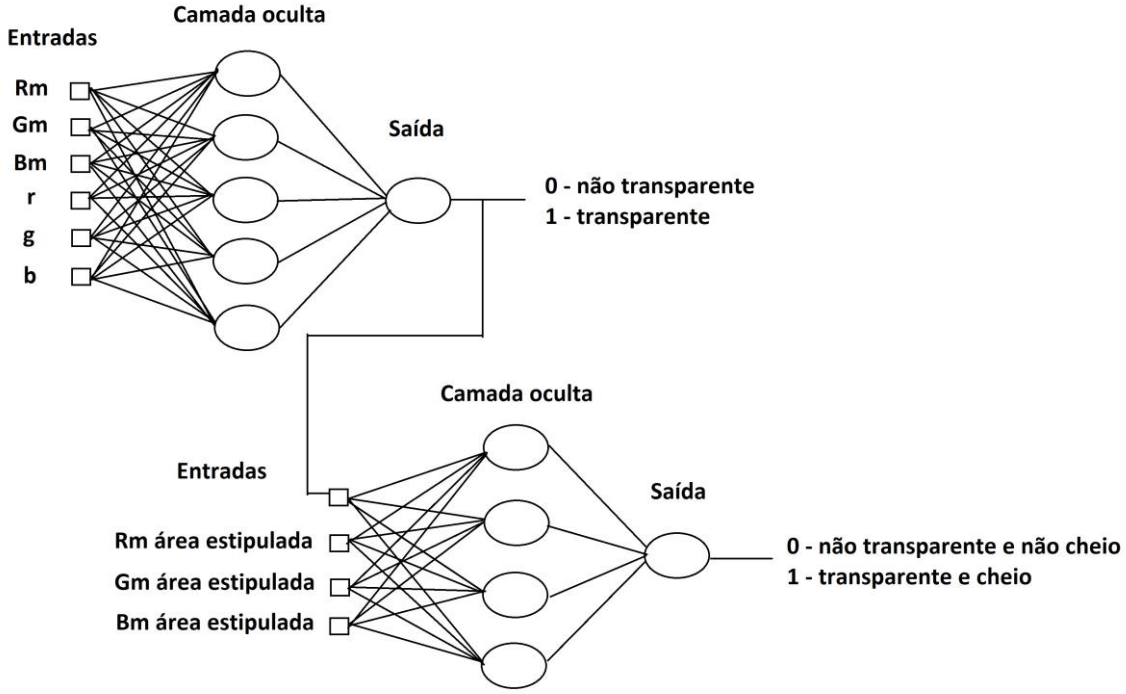

Figura 37: Rede neural implementada para classificação dos dados após tratamento especificado (Louro, et al. 2006, alterado pelo autor)

Por fim, foi realizada a classificação também empregando os métodos PCA e SVD, já tratados no capítulo de Fundamentação Teórica, para verificar qual método de pré-processamento de dados apresenta uma melhor acurácia.

4- RESULTADOS E DISCUSSÕES

Em todos os quadros apresentados neste capítulo, os itens ressaltados com cores diferentes são os que representam maior desempenho para o número de amostras e testes apresentados. Os valores de taxa de aprendizagem aplicados nos métodos SVM e RNA estão em faxas diferentes, pois foram consideradas as margens que resultaram em melhor acurácia com menor tempo de execução.

Como já tratado no capítulo anterior, primeiramente foram realizados testes utilizando o banco de dados MNIST. A Tabela 1 apresenta a relação de acurácia de todos os testes, realizando a variação do número de amostras e taxa de aprendizagem. Como o total de amostras é de 42000, elas foram divididas entre treinamento e teste. As imagens deste banco de dados já são inicialmente em *Grayscale* e possuem resolução de 28x28 pixels, portanto não houve necessidade de realizar pré-processamento.

Tabela 1: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados MNIST (próprio autor)

MNIST						MNIST					
Amostras		SVM		Feedforward		Amostras		SVM		Feedforward	
Treinamento	Testes	Learn. Rate	Acurácia	Learn. Rate	Acurácia	Treinamento	Testes	Learn. Rate	Acurácia	Learn. Rate	Acurácia
420 (1%)	41580 (99%)	0.01	9,8124	0.1	**81,51**	126000 (30%)	29400 (70%)	0.01	11,1122	0.1	**95,82**
		0.001	9,8124	0.01	71,42			0.001	11,1122	0.01	92,14
		0.0001	9,8124	0.001	27,37			0.0001	11,1122	0.001	85,29
		0.00001	10,16935	0.0001	11,27			0.00001	18,2143	0.0001	54,54
		0.000001	71,6835	0.00001	9,98			0.000001	96,0102	0.00001	13,91
		0.0000001	**87,9389**	0.000001	9,86			0.0000001	**96,4864**	0.000001	10,1
2100 (5%)	39900 (95%)	0.01	9,85965	0.1	**89,68**	16800 (40%)	25200 (60%)	0.01	11,01588	0.1	**96,23**
		0.001	9,85965	0.01	86,89			0.001	11,01588	0.01	92,9
		0.0001	9,85965	0.001	63,8			0.0001	11,01588	0.001	86,73
		0.00001	14,95238	0.0001	17,1			0.00001	18,2976	0.0001	61,23
		0.000001	89,8797	0.00001	10,46			0.000001	96,5159	0.00001	15,6
		0.0000001	**93,46366**	0.000001	9,95			0.0000001	**96,89286**	0.000001	10,23
4200 (10%)	37800 (10%)	0.01	11,1746	0.1	**92,04**	21000 (50%)	21000 (50%)	0.01	11,0095	0.1	**96,58**
		0.001	11,1746	0.01	88,88			0.001	11,0095	0.01	93,38
		0.0001	11,1746	0.001	75,49			0.0001	11,0095	0.001	87,55
		0.00001	16,4603	0.0001	26,41			0.00001	19,97143	0.0001	65,88
		0.000001	92,886	0.00001	11,04			0.000001	96,838	0.00001	17,67
		0.0000001	**94,82275**	0.000001	9,95			0.0000001	**97,133**	0.000001	10,31
8400 (20%)	33600 (20%)	0.01	11,1845	0.1	**94,64**						
		0.001	11,1845	0.01	90,85						
		0.0001	11,1845	0.001	82,29						
		0.00001	17,4583	0.0001	43,14						
		0.000001	95,2232	0.00001	12,53						
		0.0000001	**95,91369**	0.000001	10,07						

Relacionando apenas os valores obtidos neste trabalho entre os métodos SVM e RNA, é possível observar, conforme Figura 38, que com o decaimento da taxa de aprendizagem, os resultados do SVM têm uma melhora, enquanto que os da RNA têm uma grande baixa. A baixa nos valores no método RNA se dá ao fato de que todos os testes foram realizados em 10 épocas. Ao diminuir o valor da taxa de aprendizagem, será necessário um número maior de atualização dos pesos para se alcançar um valor satisfatório. Ao manter o número de épocas em 10, o algoritmo está sendo encerrado no meio do processo. Se o número de épocas aplicado fosse maior, os valores seriam próximos dos que foram apresentados pelo SVM.

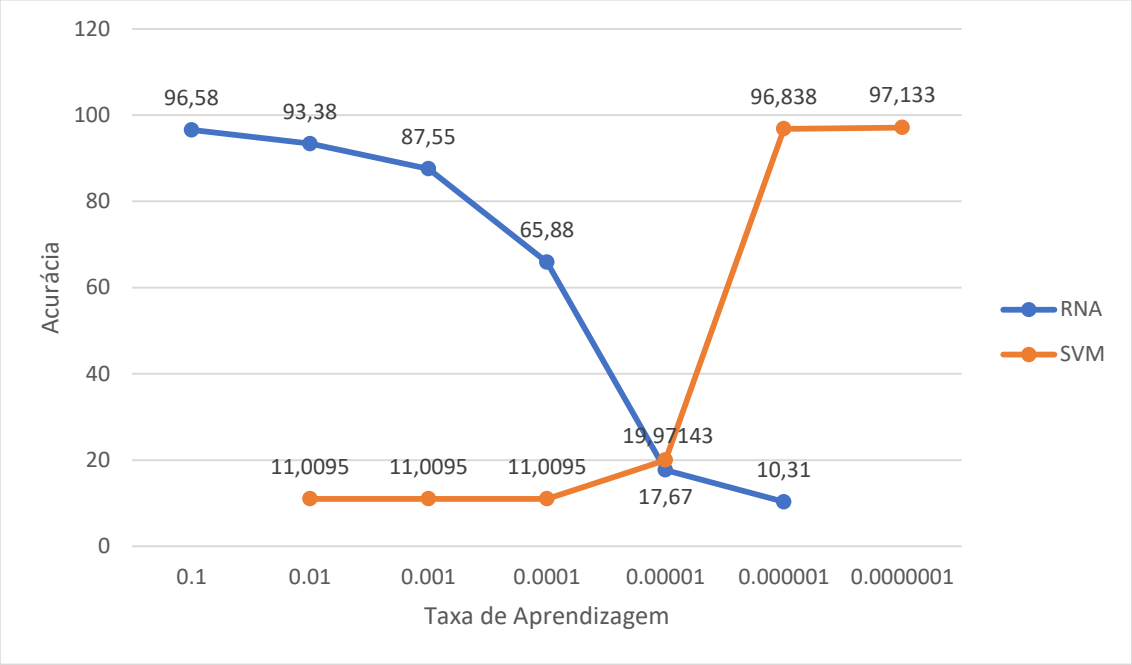

Figura 38: Relação da acurácia entre os métodos SVM e RNA para o banco de dados MNIST com 21000 amostras no treinamento e 21000 no teste (próprio autor)

É possível notar que o SVM apresenta dificuldade para realizar a classificação com valores baixos de amostras para treinamento e taxa de aprendizagem alta. Em todos os casos, apenas quando sua taxa de aprendizagem está em 0,000001 ou menor, que é obtido um resultado satisfatório. Em contrapartida, o RNA já obtem valores de acurácia interessantes com um número baixo de amostras e taxa de aprendizagem alta, porém, à medida que a taxa é diminuida, é requerido um número maior de épocas, necessitando de um maior tempo de execução.

O teste seguinte, realizado com a caneta, resultou nos valores apresentados na Tabela 2. Facilmente é observado que, para os dois métodos, quanto maior o número de amostras, maior a acurácia. O SVM se mostrou mais eficaz neste caso, apesar de apresentar resultados relevantes apenas com valores de taxa de aprendizagem de 0,0000001 ou menor. A diminuição na resolução não apresentou melhora significativa em nenhum dos dois casos. Mas, é importante ressaltar que, para um número relativamente pequeno de amostras, para uma imagem digital de um objeto real, apresentando tantas variáveis, a acurácia alcançada em ambos os casos foi expressiva. Com um maior número de amostras, seria possível obter valores consideráveis, uma vez que o mais alto obtido nos testes foi de 89,5953%.

Tabela 2: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados de uma caneta (próprio autor)

Caneta													
		192x108				96x54				48x27			
Amostras		SVM		Feedforward		SVM		Feedforward		SVM		Feedforward	
Treinamento	Testes	Learn.Rate	Acurácia	Learn.Rate	Acurácia	Learn.Rate	Acurácia	Learn.Rate	Acurácia	Learn.Rate	Acurácia	Learn.Rate	Acurácia
1110	150	0.0001	46,67	0.1	46,67	0.0001	46,67	0.1	47,33	0.0001	46,67	0.1	53,3
		0.00001	46,67	0.01	38,67	0.00001	46,67	0.01	**56,67**	0.00001	46,67	0.01	53,3
		0.000001	46,67	0.001	**53,3**	0.000001	46,67	0.001	53,33	0.000001	58	0.001	53,3
		0.0000001	54,67	0.0001	46	0.0000001	**66,67**	0.0001	46,67	0.0000001	**65,33**	0.0001	**58,67**
		0.00000001	**67,33**	0.00001	46,67	0.00000001	64,67	0.00001	46,67	0.00000001	53,33	0.00001	52,67
3263	1230	0.0001	52,0325	0.1	52,03	0.0001	52,0325	0.1	**52,03**	0.0001	52,0325	0.1	47,97
		0.00001	52,0325	0.01	46,02	0.00001	52,0325	0.01	48,05	0.00001	59,6748	0.01	41,06
		0.000001	57,6423	0.001	45,45	0.000001	72,52	0.001	50	0.000001	**73,252**	0.001	52,52
		0.0000001	**73,17**	0.0001	52,03	0.0000001	**73,008**	0.0001	**52,03**	0.0000001	72,9268	0.0001	52,03
		0.00000001	72,85	0.00001	**53,58**	0.00000001	72,85	0.00001	**52,03**	0.00000001	72,9268	0.00001	**52,6**
5263	1230	0.0001	47,9675	0.1	47,97	0.0001	47,9675	0.1	47,97	0.0001	48,0488	0.1	47,97
		0.00001	47,9675	0.01	**66,1**	0.00001	48,374	0.01	**66,1**	0.00001	81,626	0.01	**71,63**
		0.000001	61,138	0.001	57,24	0.000001	87,9675	0.001	63,01	0.000001	**89,5935**	0.001	58,21
		0.0000001	**89,1057**	0.0001	63,58	0.0000001	**89,43**	0.0001	55,28	0.0000001	80,4878	0.0001	52,2
		0.00000001	85,04	0.00001	56,83	0.00000001	74,878	0.00001	53,41	0.00000001	72,2764	0.00001	52,68

Em relação aos testes realizados com a embalagem do jogo digital, estando presente ou não o rótulo, foram obtidos os resultados presentes na Tabela 3. Com a diminuição da resolução da imagem, foi obtido um aumento de aproximadamente 6% (194 amostras) no método SVM, enquanto que no método RNA não houve melhora. Nesta aplicação o SVM também obteve melhores resultados.

Tabela 3: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados de imagens da embalagem de um jogo digital (próprio autor)

Capa Jogo													
		192x108				96x54				48x27			
		SVM		Feedforward		SVM		Feedforward		SVM		Feedforward	
Amostras		Learn. Rate	Acurácia	Learn. Rate	Acurácia	Learn. Rate	Acurácia	Learn. Rate	Acurácia	Learn. Rate	Acurácia	Learn. Rate	Acurácia
Treinamento	Testes												
3270	1868	0.0001	50,75	0.1	49,41	0.0001	49,25	0.1	49,25	0.0001	49,25	0.1	50,75
		0.00001	49,25	0.01	**63,33**	0.00001	49,25	0.01	64,35	0.00001	49,518	0.01	**63,01**
		0.000001	49,25	0.001	61,3	0.000001	50,16	0.001	62,53	0.000001	80,246	0.001	60,8
		0.0000001	58,78	0.0001	58,78	0.0000001	83,833	0.0001	**65,31**	0.0000001	91,22	0.0001	50,59
		0.00000001	**86,461**	0.00001	52,14	0.00000001	**91,488**	0.00001	65,1	0.00000001	**92,398**	0.00001	41,92

Em seguida, conforme Figura 39, é possível observar os resultados dos testes realizados no interior da embalagem e objetivando 4 saídas diferentes. Optou-se por apresentar graficamente este resultado uma vez que é visualmente aconselhável, dado que as respostas em ambas as três resoluções foram bem próximas. Uma baixa taxa de acurácia não é interessante para este projeto, que busca otimizar um processo industrial, portanto, os resultados neste teste não foram satisfatórios, uma vez que não foi alcançada a margem de 60% de acurácia. Isto se deve ao baixo número de amostras aplicadas na fase de treinamento. Foram utilizadas 2304 amostras, sendo aproximadamente 576 para cada um dos 4 casos. Resultados acima de 90% seriam adequados ao objetivo proposto.

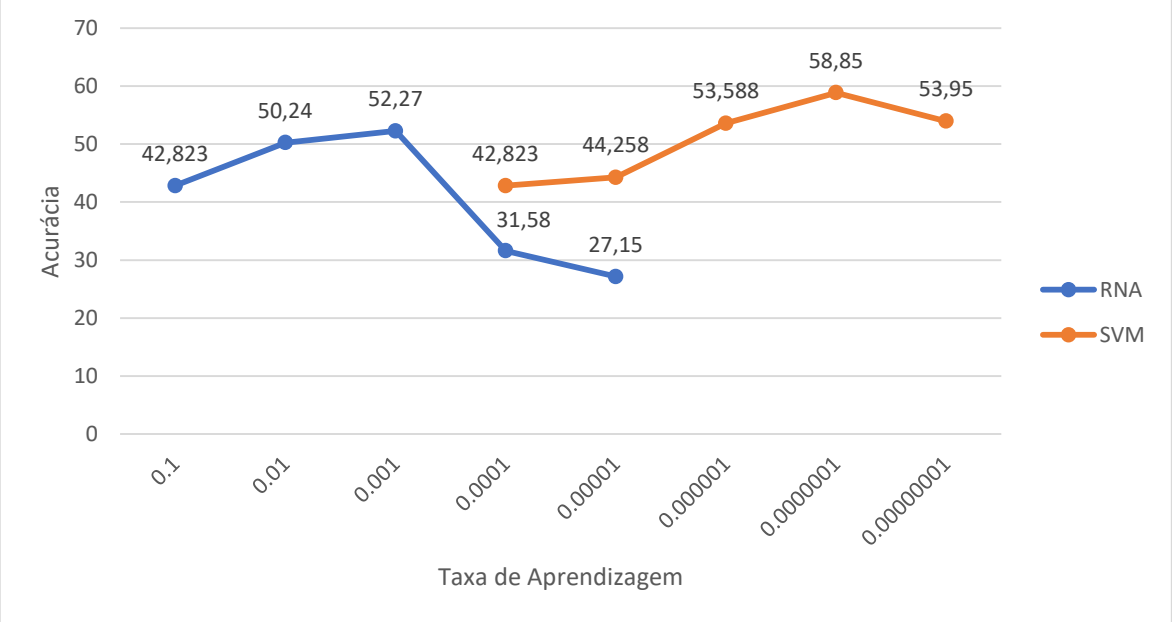

Figura 39: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados do interior da embalagem de jogo digital com 4 saídas (próprio autor)

Seguidamente, para o caso de análise do interior da embalagem do jogo objetivando uma classificação de 2 saídas, onde o interior possui algum item (sendo o manual e/ou o cartucho) ou está vazio, foi obtido o resultado da Figura 40. Da mesma forma, preferiu-se apresentar o produto deste teste graficamente, uma vez que os resultados em ambas as têS resoluções foram próximos. Neste caso, o algoritmo RNA demonstrou um melhor resultado, enquanto que o SVM começou a ter uma melhor aplicabilidade com valores mínimos de taxa de aprendizagem. Com valores maiores de época, seria possível alcançar uma acurácia melhor com o RNA, porém exgiria maior tempo de execução.

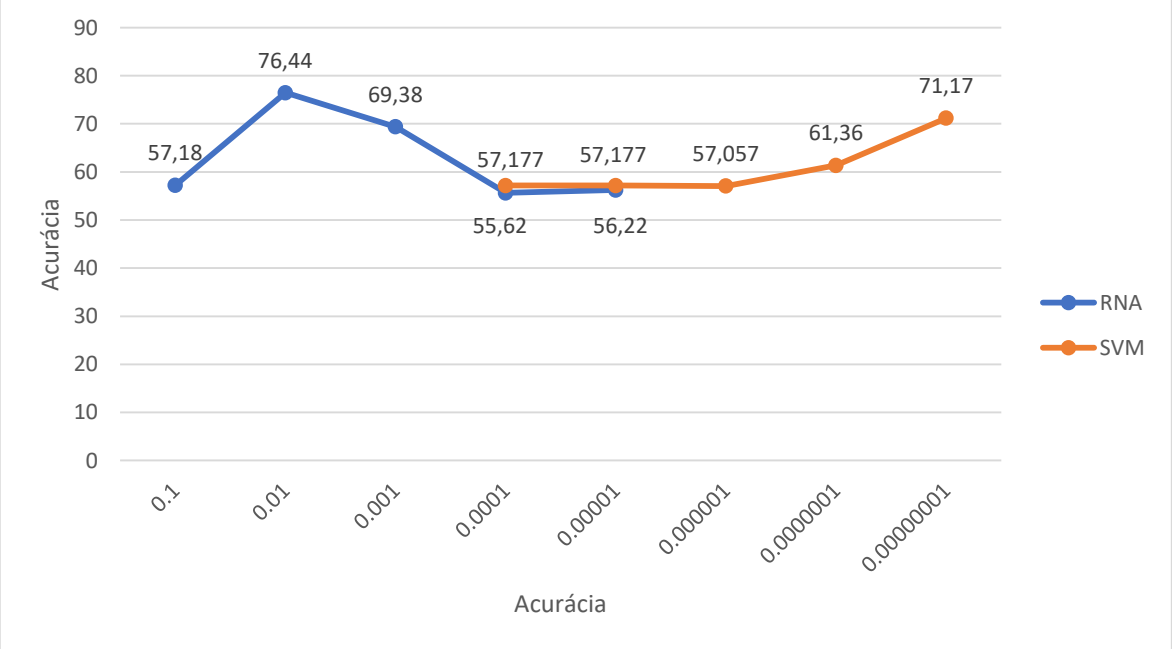

Figura 40: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados do interior da embalagem de jogo digital com 2 saídas (próprio autor)

O teste do interior da embalagem – estando apenas o manual e o cartucho ou estando vazio (resposta binária) – apresentou excelentes resultados, conforme Tabela 4. A acurácia máxima encontrada em ambos os métodos foi de 100% e, quanto menor a resolução, maior a precisão, o que prova que, para este caso, os detalhes não são importantes e são fatores que obstruem a obtenção de uma melhor resposta.

Tabela 4: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados do interior da embalagem de jogo digital estando vazia ou cheia (próprio autor)

Interior Capa - Cheio e Vazio									
		480x270				192x108			
		SVM		Feedforward		SVM		Feedforward	
Amostras		Learn. Rate	Acurácia	Learn. Rate	Acurácia	Learn. Rate	Acurácia	Learn. Rate	Acurácia
Treinamento	Testes								
1187	471	0.0001	76,0085	0.1	76	0.0001	76,0085	0.1	76
		0.00001	76,0085	0.01	77,28	0.00001	76,0085	0.01	76
		0.000001	76,0085	0.001	**97,45**	0.000001	76,0085	0.001	**96,6**
		0.0000001	76,0085	0.0001	93,63	0.0000001	86,84	0.0001	71,13
		0.00000001	**87,47**	0.00001	82,8	0.00000001	**99,575**	0.00001	64,5
		96x54				48x27			
		SVM		Feedforward		SVM		Feedforward	
Amostras		Learn. Rate	Acurácia	Learn. Rate	Acurácia	Learn. Rate	Acurácia	Learn. Rate	Acurácia
Treinamento	Testes								
1187	471	0.0001	76,0085	0.1	76	0.0001	76,0085	0.1	76
		0.00001	76,0085	0.01	**100**	0.00001	77,5	0.01	**100**
		0.000001	77,707	0.001	95,11	0.000001	90,23	0.001	84,29
		0.0000001	94,48	0.0001	81,74	0.0000001	**100**	0.0001	60,3
		0.00000001	**100**	0.00001	64,97	0.00000001	**100**	0.00001	76

Para o caso dos 4 bonecos diferentes, foi obtido o resultado apresentado na Figura 41. Da mesma forma que com os casos anteriores, não houve muita variação da acurácia na mudança de resolução, por isto a resposta é mostrada graficamente, apresentando um soluto de todos os valores. Nesta aplicação o RNA obteve resultado significativamente maior, dintinguindo os objetos de forma eficaz.

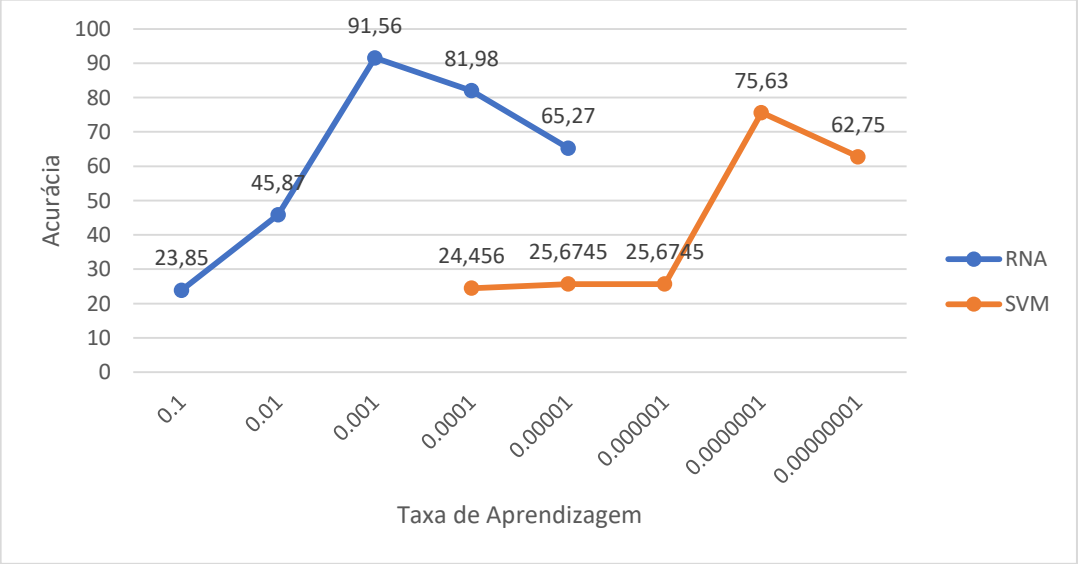

Figura 41: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados dos 4 bonecos distintos (próprio autor)

Por fim, foram realizados os testes com um único boneco, classificando-a quanto à presença ou ausência dos braços. A Figura 42 apresenta o resultado desta classificação, onde o método SVM expressou melhor resultado com valor máximo de acurácia alcançado de 88,95%.

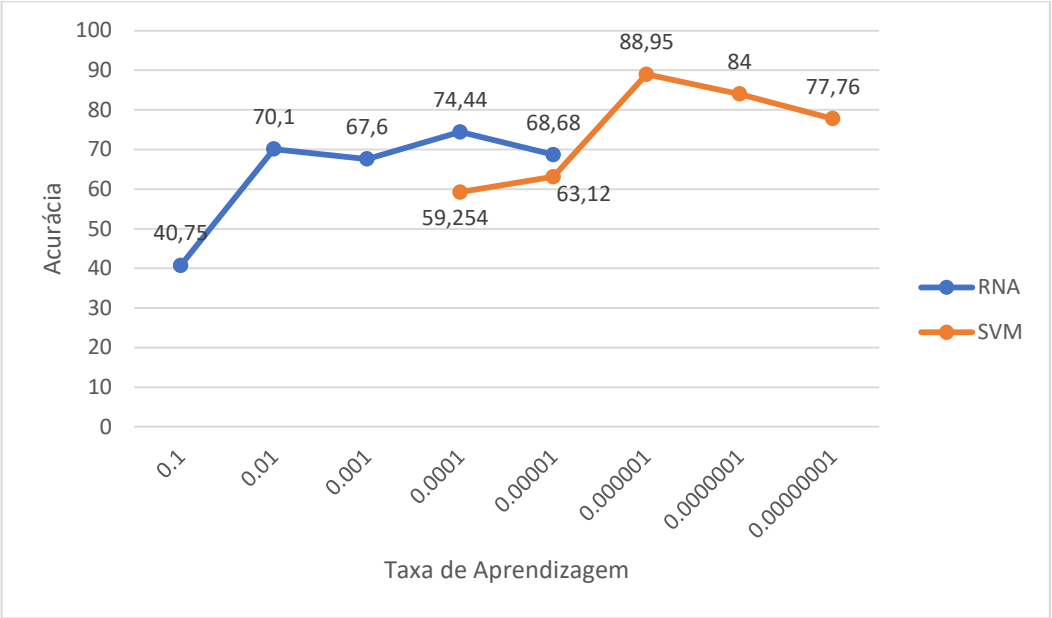

Figura 42: Relação de acurácia para cada método a partir do número de amostras para treinamento e teste e taxa de aprendizagem para o banco de dados de um boneco (próprio autor)

A partir dos dados observados pelos resultados já apresentados, foram realizadas melhoria, como o desenvolvimento de uma GUI para melhor acompanhamento dos parâmetros e respostas pelo usuário, uma vez que tal acompanhamento sendo realizado diretamente por linhas de códigos não é viável caso o usuário não tenha conhecimentos de programação. Também, a implementação de outros métodos de tratamento e processamento de dados foi importante para redução do banco de dados.

A segunda etapa do projeto foi a realização da classificação de refis de canetas para aplicação de insulina. Um mesmo objeto foi fotografado diversas vezes, de forma a possuir variações a cada imagem adquirida, com coloração e volume diferentes. A insulina de cor aceitável é incolor, sendo qualquer outra entrada considerada inadmissível. Da mesma forma, apenas refis com volume total seriam tolerados. Seis bancos de dados foram utilizados para treinamento, sendo que, em 4, a câmera foi posicionada à 8,5cm de distância do objeto e, em

2, à uma distância mais próxima, deixando o refil em escala maior na imagem. Outros dois bancos de dados foram utilizados para a fase de teste, com imagens diferentes das apresentadas na fase de treinamento. A tabela 5 apresenta a relação de imagens em cada banco de dados.

Tabela 5: Relação de amostras nos bancos de dados (próprio autor)

Treinamento 1		Treinamento 4	
Transparente	6	Transparente	61
Não transparente	34	Não transparente	94
Cheio	6	Cheio	9
Não cheio	34	Não cheio	146
Total	40	**Total**	155
Treinamento 2		Treinamento 5	
Transparente	58	Transparente	12
Não transparente	22	Não transparente	18
Cheio	6	Cheio	7
Não cheio	74	Não cheio	23
Total	80	**Total**	30
Treinamento 3		Treinamento 6	
Transparente	26	Transparente	36
Não transparente	94	Não transparente	48
Cheio	6	Cheio	10
Não cheio	114	Não cheio	74
Total	120	**Total**	84
Teste 1		Teste 2	
Transparente	12	Transparente	3
Não transparente	52	Não transparente	7
Cheio	6	Cheio	3
Não cheio	58	Não cheio	7
Total	64	**Total**	10

Para cada banco de dados foi realizada a classificação através dos algoritmos de RNA e SVM, sendo tratados por PCA, SVD ou segmentação da imagem. Também foram observados os tempos de execução do algoritmo para as etapas de treinamento e teste para diferentes escalas, conferindo assim sua viabilidade computacional. Utilizando, ao banco de dados de treinamento 1 e ao de testes 1, o algoritmo RNA com processamento para redução de dados pelo método SVD com escala de 1:10, foram necessários 21 segundos para a fase de treinamento e 33 segundos para a fase de teste. Para a escala 1:5, 1 minuto e 2 segundos para o treinamento e 1 minuto e 40 segundos para o teste. Em escala 1:2, 6 minutos e 19 segundos e 37 minutos e 52 segundos, respectivamente.

Notou-se ser inviável a aplicação à escala 1:1, uma vez que a acurácia do algoritmo apresentava variações mínimas em relação à escala para os primeiros três testes, com valores em torno de 76% e 81%. Desta forma, foi definido aplicar apenas a escala 1:10.

Todos os parâmetros aplicados ao treinamento – como taxa de aprendizagem, alfa, etc – foram definidos de forma empírica, verificando quais resultavam em melhor acurária à todos os métodos e dados como padrão em todos os testes. Assim, verifica-se a dinâmica de cada algoritmo e tratamento de forma ponderada.

Para o primeiro caso, conforme Tabela 6, os resultados foram muito bons para testes realizados com os dados utilizados no treinamento da própria RNA, porém, não foram satisfatórios para dados diferentes. Apenas o banco de dados com 120 amostras apresentou resultado aceitável. Tal resultado se deve ao maior número de imagens na fase de treinamento.

Tabela 6: Relação de acurácia (%) no treinamento por RNA e tratamento SVD para as características de coloração e volume (próprio autor)

RNA - Tratamento SVD						
Nº	Ao próprio treinamento			Ao teste		
	Coloração (%)	Volume (%)	Tempo	Coloração (%)	Volume (%)	Tempo
40	100	100	21s	73,44	76,56	33s
80	100	93,75	45s	78,12	89,06	37s
120	100	95,83	1min20s	82,81	87,5	37s
155	100	94,84	1min32s	57,81	85,94	36s
30	100	86,67	18s	50	50	5s
84	100	88,1	50s	50	70	6s

Em seguida, pelo tratamento PCA, foi realizado o treinamento RNA, conforme Tabela 7, em todos os bancos de dados. Para este caso também foram apresentados resultados satisfatórios diretamente proporcionais à quantidade de dados.

Tabela 7: Relação de acurácia (%) no treinamento por RNA e tratamento PCA para as características de coloração e volume (próprio autor)

RNA – Tratamento PCA						
Nº	Ao próprio treinamento			Ao teste		
	Coloração (%)	Volume (%)	Tempo	Coloração (%)	Volume (%)	Tempo
40	100	100	27s	75	84,38	36s
80	100	93,75	44s	84,38	87,5	34s
120	100	95,83	1min06s	92,19	87,5	36s
155	100	94,84	1min25s	79,69	90,62	36s
30	100	86,67	22s	30	60	8s
84	100	88,1	57s	50	60	6s

O resuldado da aplicação do tratamento de segmentação da imagem colorida aos bancos de dados para treinamento pelo algoritmo RNA pode ser observado na Tabela 8. Apesar deste método apresentar resultados mais baixos em relação aos dados do próprio treinamento, ele apresentou resultados mais satisfatório ao teste, mantendo certa estabilidade aos valores, independente da quantidade de amostras no banco de dados. Isto se deve, provavelmente, porque o algoritmo não apresentou *overfitting*, ou seja, não houve uma adaptação ao banco de dados apresentado no treinamento, não havendo a perda da capacidade de generalização. Uma maior variedade de dados poderia ocasionar em uma melhora da acurácia.

Tabela 8: Relação de acurácia (%) no treinamento por RNA e tratamento de segmentação para as características de coloração e volume (próprio autor)

RNA – Tratamento Segmentação						
Nº	Ao próprio treinamento			Ao teste		
	Coloração (%)	Volume (%)	Tempo	Coloração (%)	Volume (%)	Tempo
40	85	100	1min10s	73,44	85,94	1min52s
80	82,5	93,75	2min23s	73,44	87,5	1min52s
120	80	95,83	3min36s	81,25	89,06	1min52s
155	72,26	94,84	4min49s	73,44	89,06	1min50s
30	66,67	86,67	55s	60	70	17s
84	88,71	88,1	2min40s	90	70	18s

O próximo passo foi realizar os devidos treinamentos e testes utilizando o algoritmo SVM. A Tabela 9 apresenta os resultados para o tratamento SVD, onde

nota-se certa estabilidade nos valores de acurácia. Este resultados serão tratados mais adiante.

Tabela 9: Relação de acurácia (%) no treinamento por SVM e tratamento SVD para as características de coloração e volume (próprio autor)

SVM – Tratamento SVD						
Nº	Ao próprio treinamento			Ao teste		
	Coloração (%)	Volume (%)	Tempo	Coloração (%)	Volume (%)	Tempo
40	100	100	22s	81,25	90,62	32s
80	100	100	44s	81,25	95,31	32s
120	100	100	1min10s	81,25	95,31	32s
155	100	100	1min27s	81,25	95,31	33s
30	100	100	16s	60	70	5s
84	100	100	52s	60	60	6s

Em seguida, para o tratamento PCA, foram obtidos os resultados apresentadados na Tabela 10 e que também serão abordados posteriormente.

Tabela 10: Relação de acurácia (%) no treinamento por SVM e tratamento PCA para as características de coloração e volume (próprio autor)

SVM – Tratamento PCA						
Nº	Ao próprio treinamento			Ao teste		
	Coloração (%)	Volume (%)	Tempo	Coloração (%)	Volume (%)	Tempo
40	100	100	21s	81,25	90,62	33s
80	100	100	44s	81,25	95,31	33s
120	100	100	59s	81,25	95,31	32s
155	100	100	1min18s	81,25	95,31	32s
30	100	100	17s	60	70	6s
84	100	100	44s	60	60	5s

Os resultados para o tratamento por segmentação da imagem colorida e treinamento por SVM podem ser observados na Tabela 11. Para este caso foi obtida acurácia de 100% em relação à coloração para as amostras do banco de dados des testes 2. Além disso, foram obtidos valores interessantes para a saída de análise da coloração e do volume, obtendo apenas uma previsão errada em relação às 64 amostradas.

Tabela 11: Relação de acurácia (%) no treinamento por SVM e tratamento de segmentação para as características de coloração e volume (próprio autor)

SVM – Tratamento Segmentação						
Nº	Ao próprio treinamento			Ao teste		
	Coloração (%)	Volume (%)	Tempo	Coloração (%)	Volume (%)	Tempo
40	100	100	1min10	60,94	93,75	1min53
80	86,25	100	2min20	75	98,44	1min53
120	82,5	100	3min30s	78,12	98,44	1min53s
155	83,23	100	4min50s	73,44	96,88	1min53s
30	86,67	100	55s	70	70	19s
84	82,14	94,05	2min32s	100	60	18s

Se tratando dos resultados apresentados nos quadros 12 e 13, é possível notar a persistência do valor 81,25%. Este resultado não é considerado aceitável uma vez que representa exatamente o número de amostras não transparentes presentes no banco de dados. Se tratando da classificação em relação à coloração, para amostras transparentes, é dado o valor 1 e para a turvas ou coloridas, é dado o valor 0. O que acontece neste caso é que o algoritmo retorna como resposta para todos os casos o valor 0. Ou seja, ele acredita que todas as amostras são inválidas. Neste caso, é apresentado o problema de *overfitting*, em que o algoritmo obteve a perda de generalização, acreditando que em todos os casos, não há refil transparente. A grande diferença na quantidade de amostras consideradas corretas em relação às consideradas incorretas pode ser a causa deste problema. De forma geral, resultados de coloração para as imagens de teste com valores iguais ou menores que 81,25% podem ser descartados. Assim sendo, o algoritmo RNA com tratamento por PCA e SVD obtiveram melhor resultado para a classificação da coloração.

Tratando do problema de *overfitting*, com base nos testes realizados, observou-se que algumas abordagens podem ser tomadas, como o aumento do banco de dados priorizando a utilização de imagens diferentes e a redução do tamanho da rede neural artificial.

Para o caso da classificação de volume e coloração, os testes realizados pelo método SVM apresentaram maior valor de acurácia.

Foi desenvolvida também, apenas por curiosidade, uma única estrutura de classificação empregando o método RNA ou SVM em que houvesse como entrada todos os parâmetros para coloração e todos para volume. A melhor acurácia encontrada para este caso foi de 96,875%.

Também foi desenvolvida uma estrutura aplicando o método RNA, inicialmente, para classificação da coloração, em série à uma classificação de volume pelo método SVM, onde as entradas desta segunda seriam os parâmetros para definição do volume e a saída da classificação da coloração, de forma a obter como resposta 1 todo refil com volume e coloração ideais e 0 qualquer saíde diferente desta. Para este caso, foram utilizados os parâmetros de melhor resultado para a coloração, que foi o caso do algoritmo RNA com pré-processamento por PCA e banco de dados de treinamento de 120 amostras, e um dos parâmetros de melhor resultado para o volume, que foi o SVM, também com pré-processamento por PCA e banco de dados de treinamento de 120 amostras. Ao aplicar esta estrutura em série com os melhores resultados para ambas as classificações, foi obtida acurácia de 100% ao banco de dados de testes de 64 amostras, para classificação de refis.

Por fim, em relação a todos os resultados obtidos, é possível destacar os seguintes pontos:

• O número de épocas aplicado ao método RNA define radicalmente a acurácia alcançada nos testes. Se o número de épocas definido não for suficientemente inversamente proporcional à taxa de aprendizagem aplicada, o algoritmo não irá alcançar seu máximo desempenho. Quanto menor a taxa de aprendizagem, maior o número de épocas necessárias para um melhor desempenho.

• A quantidade de amostras na fase de treinamento é importante para uma melhor acurácia. Quanto maior o número de amostras e sua diversidade, melhor a eficiência do algoritmo para identificação de imagens e menor a chance de perda de generalização, evitando o problema de *overfitting*.

• Para o caso dos testes realizados neste projeto, é possível obter valores superiores de acurácia aplicando um número maior de amostras.

• É importante realizar um estudo de caso para cada objeto, uma vez que alguns detalhes podem ser descartados durante a análise. No teste realizado com o interior da capa do jogo digital, objetivando classificar apenas se está cheio ou vazio, é possível notar que, ao diminuir a resolução, os resultados obtidos apresentam relevante melhora. A presença de detalhes neste tipo de classificação não é um fator significativo, pois, com a embalagem vazia, seu interior é totalmente preto. Em contrapartida, no caso da caneta, a presença de detalhes é importante, uma vez que, como a caneta e sua tampa são objetos finos, ao

diminuir a resolução, diversos pixels importantes podem ser perdidos, dificultando a classificação.

• Para classificações com mais de 2 saídas, como é o caso da análise dos 4 bonecos distintos, o RNA apresenta melhor resultado.

• A variedade de amostras na fase de treinamento influencia diretamente ao resultado da previsão, uma vez que o número de amostras para uma saída 1 pode ser muito inferior aos da saída 0, dificultando a análise.

• A definição dos parâmetros referentes aos testes varia de acordo com o banco de dados a ser analisado. Deve-se realizar diversos ensaios e estudos de caso para a definição de quais parâmetros atuam melhor sobre o problema. Inicialmente, as previsões sempre resultavam em valor 0 devido à parâmetros estabelecidos incorretamente.

• A aplicação de métodos de pré-processamento e tratamento de dados é uma opção interessante para a definição de custos temporais e computacionais. Em alguns casos, em que não foi realizada a aplicação, houve elevada perda de tempo, com treinamentos com mais de uma hora de duração. Foi possível observar que, aplicando tais métodos, pode-se reduzir este tempo para poucos minutos, sem a perda de informações importantes à classificação.

5- CONCLUSÕES

Conlui-se que foi obtido um resultado satisfatório. Por se tratar de um tema que não é muito abordado no curso, foi um grande desafio a ser enfrentado. Para um primeiro estudo acerca do assunto, foi de grande valia, uma vez que não se limitou apenas ao campo das redes neurais artificiais e aprendizagem de máquinas, mas também como o de métodos de pré-processamento de dados, redução de matrizes de características e programação em linguagem *Python* voltada à GUI.

5.1- PROPOSTAS PARA TRABALHOS FUTUROS

Um ponto importante que não foi possível tratar durante o desenvolvimento deste trabalho foi a implementação de identificação de deformidades em tempo real, através da utilização de câmeras.

Outra abordagem seria a implementação conjunta de RNAs e SVMs. No caso dos resultados obtidos, a coloração do refil foi melhor identificada pela RNA e o volume pela SVM, então, colocando os dois algoritmos trabalhando em conjunto sequencialmente, talvez possa ser gerado um resultado distinto.

6- REFERÊNCIAS BIBLIOGRÁFICAS

ABREU, R. A. **Perdas no processo produtivo**. RAA Consultoria: 2002.

ANTONELLO, Ricardo. **Introdução a Visão Computacional com Python e OpenCV**. Parte 1 de 2. Versão 0.1a – Não corrigida. IFC – Campus Luzerna, 2017.

ARAÚJO JÚNIOR, Antônio P. de; CHAGAS, Christiano V. de; FERNANDES, Raphaela G. **Uma rápida análise sobre automação industrial**. Redes para Automação Industrial. DCA-CT-UFRN. 2003.

AUGUSTO, Karen Soares; PACIORNIK, Sidnei; GOMES, Otávio. **Identificação Automática do Grau de Maturação de Pelotas de Minério de Ferro**. Rio de Janeiro, 2012. 183p. Dissertação de Mestrado - Departamento de Engenharia de Materiais, Pontifícia Universidade Católica do Rio de Janeiro.

BARANAUSKAS, J. A; MONARD, M. C. (2000). ***Reviewing some machine learning concepts and methods***. *Technical Report* 102, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/ rel_tec/RT_102.ps.zip.

BARRETO, Jorge M. **Introdução às Redes Neurais Artificiais**. Anais V Escola Regional de Informática da SBC Regional Sul, 5 a 10 de maio de 1997. Páginas 41 - 71.

BARRETO, Jorge M. **Introdução às Redes Neurais Artificiais**. Laboratório de Conexionismo e Ciências Cognitivas UFSC -Departamento de Informática e de Estatística 88040-900 - Florianópolis - SC. 2002.

BATISTA, Gustavo E. A. P. A. **Pré-processamento de Dados em Aprendizado de Máquina Supervisionado**. Tese de Doutorado – ICMC-USP. 2003.

BURGES, C. J. C. (1998). ***A tutorial on support vector machines for pattern recognition***. *Knowledge Discovery and Data Mining*, 2(2):1–43.

BOCANEGRA, Charlie W. R; SILVA, Antônio N. R. da. **Procedimentos para tornar mais efetivo o uso das redes neurais artificiais em planejamento de transportes**. Dissertação de Mestrado. Universidade de São Paulo. São Carlos. 2002.

CALIXTO, R. R; ARAGÃO, M. F; RODRIGUES, A. B; NETO, L. G. P; CAVALCANTE, T. S. **Sistema de visão computacional para classificação de melão amarelo de acordo com o formato**. XXXIV SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES – SBrT 2016, 30 DE AGOSTO A 02 DE SETEMBRO, SANTARÉM, PA.

CRISTIANINI, N; SHAWE-TAYLOR, J. (2000). ***An Introduction to Support Vector Machines and other kernel-based learning methods***. *Cambridge University Press.*

DENG, J; DONG, W; SOCHER, R; LI, Li-Jia; LI, K; FEI-FEI, L. ***Imagenet: A large-scale hierarchical image database***. *In Computer Vision and Pattern Recognition*, 2009. CVPR 2009. *IEEE Conference on*. IEEE, 2009, pp. 248–255.

FERG, Stephen. ***Thinking in Tkinter***. 2009. Disponível em: http://thinkingtkinter.sourceforge.net/

FERNEDA, Edberto. **Redes neurais e sua aplicação em sistemas de recuperação de informação**. Ci. Inf. [online]. 2006, vol.35, n.1, pp.25-30. ISSN 0100-1965.

FILHO, Edson Costa de Barros Carvalho. **Modelagem, Aplicações e Implementações de redes Neurais**. Anais da IV Escola Regional de Informática da SBC Regional Sul, 21 a 27 de abril de 1996. Páginas 36 - 53.

FLECK, L; TAVARES, M.H.F; EYNG, E; HELMANN, A. C; ANDRADE, M. A. de Moares. **REDES NEURAIS ARTIFICIAIS: PRINCÍPIOS BÁSICOS**. Revista Eletrônica Científica Inovação e Tecnologia 7 (13), 47-57. 2016.

GARCIA-PENA, Marisol; ARCINIEGAS-ALARCON, Sergio; BARBIN, Décio. **Imputação de dados climáticos utilizando a decomposição por valores singulares: uma comparação empírica**. Rev. bras. meteorol., São Paulo , v. 29, n. 4, p. 527-536, Dec. 2014 .

GHAZANFARI, A; IRUDAYARAJ, J; Kusalik, A. ***Grading pistachio nuts using a neural network approach***. Transactions of the ASAE 1996. Vol.39(6): 2319-2324.

GIBNEY, E. ***Google ai algorithm masters ancient game of go***. *Nature*, vol. 529, pp. 445–446, 2016.

GITBUB. ***CS231n Convolutional Neural Networks for Visual Recognition***. Acessado em: 20/05/2018. Disponível em: http://cs231n.github.io/neural-networks-2/

GITHUB. ***How To Train an Object Detection Classifier for Multiple Objects Using TensorFlow (GPU) on Windows 10***. Acessado em 25/04/2018. Disponível em: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10.

GORNI, Antônio Augusto. **Redes Neurais Artificiais - Uma Abordagem revolucionária em Inteligência Artificial**. Revista MicroSistemas edição 133 páginas 14 a 25 e edição 134 páginas 14 a 17, Ano XII.

GRANATYR, Jones. **Redes Neurais Artificiais em Python**. Curso On-Line. Plataforma Udemy. 2017.

GUPTA, Madan M; RAO, Dandina H. **Neuro-Control Systems**. Um volume selecionado reeditado. *IEEE Neural Networks Council*, Sponsor.

HAYKIN, S. (1999). ***Neural Networks - A Compreensive Foundation***. *Prentice-Hall, New Jersey, 2 edition.*

HEBB, Donald. ***The Organization of Behavior***. 1949.

HUMULIN®: Insulina humana (derivada de ADN* recombinante). Indústria Lilly France S.A.S. Distribuído por Eli Lilly do Brasil Ltda. CRF-SP 11422. Bula de remédio on-line. Disponível em: http://www.saudedireta.com.br/catinc/drugs/bulas/humulin.pdf

JAIN, R.; KASTURI, R.; SCHUNCK, G. B. (1995). ***Machine Vision***. *McGraw-Hill, first Edition.*

KERAS. Disponível em: <https://keras.io/>. Acesso em: 24/04/2018.

KRIZHEVSKY, A; SUTSKEVER, I; HINTON, G. E. ***Imagenet classification with deep convolutional neural networks***. *In Advances in neural information processing systems*, 2012, pp. 1097–1105.

LECUN, Y; BOTTOU, L; BENGIO, Y; HAFFNER, P. ***Gradient-based learning applied to document recognition***. *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998

LORENA, Ana Carolina; CARVALHO, André C. P. L. F. de. ***Introdução às Máquinas de Vetores Suporte (Support Vector Machines)***. Relatórios Técnicos do ICMC. São Carlos. 2003.

LOURO, Antônio Henrique Figueira; MENDONÇA, Michelle Magalhães; GONZAGA, Adilson. **Classificação de tomates utilizando redes neurais artificiais**. Anais. São Carlos: EESC-USP, 2006.

MARTINS, Mauricio; GUIMARÃES, Lamartine; MARIA, Leila; FONSECA, Garcia. **Classificador de texturas por redes neurais**. 2018.

MATURANA, Patrícia S. **Algoritmos de detecção de bordas implementados em FPGA**. Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2010

MCCULLOCH, Warren; PITTS, Walter. ***A Logical Calculus of the Ideas Immanent in Nervous Activity***. *Bulletin of Mathematical Biophysics*, 1943.

MITCHELL, T. (1997). ***Machine Learning***. *McGraw Hill.*

NAKANO, K. ***Application of neural networks to the color grading of apples.*** Computers and Electronics in Agriculture 18 (1997) 105-116.

NOGUEIRA, Ricardo C. D. A. **Análise de Conversão de Imagem Colorida para Tons de Cinza Via Contraste Percebido**. Trabalho de Graduação em Engenharia da Computação. Universidade Federal de Pernambuco. 2016. [Orientador: Prof. Dr. Carlos Alexandre Barros de Mello]

NOVORAPID® FLEXPEN®: insulina asparte. Fabricado por: Novo Nordisk A/S DK-2880, Bagsvaerd, Dinamarca. Importado por: Novo Nordisk Farmacêutica do Brasil Ltda. Farmacêutico responsável: Luciane M. H. Fernandes CRF/PR n°6002. Esta bula foi aprovada pela Anvisa em 17/03/2017. Disponível em: http://www.novonordisk.com.br/content/dam/brazil/affiliate/www-novonordisk-br/Bulas/NovoRapid%20FlexPen%20Paciente.pdf

OLIVEIRA, Juliano V. **Estudo da Decomposição em valores Singulares e Análise Dos Componentes Principais**. Trabalho de conclusão do Curso de Matemática. Universidade Federal Fluminense. Volta Redonda. Agosto de 2016. [Orientador: Profa. Dra. Marina Sequeiros Dias Freitas]

SANCHES, Carlos H.; FONTOURA, Paulo J.; VIERA, PHILLYPI F.; BATISTA, Marcos A. **Técnicas de Suavização de Imagens e Eliminação de Ruídos**. Anais do EATI. Frederico Westphalen - RS. Ano 5 n. 1. p. 21-30. Nov/2015.

STIVANELLO, Maurício. E. **Inspeção Industrial através de Visão Computacional.** Trabalho de Conclusão de Curso do curso de Bacharel em Ciências da Computação, Universidade Regional de Blumenau. 2004.

PORTES DE ALBUQUERQUE, Marcio & Marcelo de. **Processamento de Imagens: Métodos e Análises**, Revista de Ciência e Tecnologia (ISSN 1519-8022), vol1 – n.1 – pg.10-22 - FaCET(2000).

RAVINDRA, Savaram. ***How Convolutional Neural Networks Accomplish Image Recognition?***. Artigo digital originalmente publicado em 2017 na KDnuggets. Acessado em 06/06/2018. Disponível em: https://www.kdnuggets.com/2017/08/convolutional-neural-networks-image-recognition.html

ROBLES JÚNIOR, A. **Custos de qualidade; uma estratégia para a competição global**. São Paulo: Atlas. 1994.

ROCHA, Rafael; MILIDIÚ, Ruy Luiz. **Reconhecimento de Objetos por Redes Neurais Convolutivas**. Rio de Janeiro, 2015. 48p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

RUMELHART, David E; MCCLELLAND, James L. ***Parallel Distributed Processing: Explorations in the Microstructure of Cognition (vol.1: Foundations, vol.2: Psychological and Biological Models)***. 1986.

RUSSEL AND NORVIG (1995). ***Artificial Intelligence - a Modern Approach***. *Prentice Hall.*

SANTO, Rafael do Espírito. **Utilização da Análise de Componentes Principais na compressão de imagens digitais**. Einstein (São Paulo), São Paulo , v. 10, n. 2, p. 135-139, June 2012.

SCIKIT LEARN. ***Support Vector Machines***. Ferramenta da Linguagem de Programação Python. Acessado em: 22/05/2018. Disponível em: http://scikit-learn.org/stable/modules/svm.html

SILVA, G; MARTINS, J; BARBOSA, Jair A. **JImageResizer - Um Redimensionador de Imagens feito em Java**. Trabalho desenvolvido como projeto de pesquisa no curso Bacharelado em Ciência da Computação da Universidade Católica de Brasília. 2011.

TENSORFLOW. Disponível em: <https://www.tensorflow.org/>. Acesso em: 24/04/2018.

VAPNIK, V. N. (1982). ***Estimation of dependencies based on empirical data***. *SpringerVerlag, New York.*

VERT, J.-P. (2001). ***Introduction to support vector machines and applications to computational biology (draft)***. http://web.kuicr.kyoto-u.ac.jp/~vert/research/semsvm/.

ZIEN, A; RATSCH, G; MIKA, S; SCHOLKOPF, B; LENGAEUER, T; MULLER, K. R. (2000). ***Engineering suppot vector machine kernels that recognize translation initiation sites in DNA***. *Bioinformatics*, 16:906–914.