



FACULDADE ÚNICA DE IPATINGA

**GEISEL SAMUEL GONÇALVES
IGOR FERNANDES OLIVEIRA
MARCOS PAULO REIS DO CARMO
RAFAEL RAYRON GONÇALVES**

**DETECÇÃO DE PESSOAS USANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL (IA)
COM TERMOGRAFIA.**

**IPATINGA
2019**

**GEISEL SAMUEL GONÇALVES
IGOR FERNANDES OLIVEIRA
MARCOS PAULO REIS DO CARMO
RAFAEL RAYRON GONÇALVES**

**DETECÇÃO DE PESSOAS USANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL (IA)
COM TERMOGRAFIA.**

Trabalho de Conclusão de Curso apresentado ao
curso de Ciência da Computação da Faculdade
Única de Ipatinga como requisito parcial para
obtenção ao título de bacharel em Ciência da
Computação

Orientador(a): Júlio Cesar da Silva Costa

**IPATINGA
2019**

FACULDADE ÚNICA DE IPATINGA

**GEISEL SAMUEL GONÇALVES
IGOR FERNANDES OLIVEIRA
MARCOS PAULO REIS DO CARMO
RAFAEL RAYRON GONÇALVES**

DETECÇÃO DE PESSOAS USANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL (IA) COM TERMOGRAFIA.

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação da Faculdade Única de Ipatinga como requisito parcial para obtenção ao título de bacharel em Ciência da Computação

Aprovada em 13 / 12 / 2019

BANCA EXAMINADORA

Professor: Júlio Cesar da Silva Costa
Faculdade Única de Ipatinga

Professor: Jutson Ribeiro Machado
Faculdade Única de Ipatinga

Professor: Márcio Azevedo Reis Ribeiro
Faculdade Única de Ipatinga

AGRADECIMENTOS

A Deus por ter nos dado saúde e força para superar as dificuldades.

A esta universidade, o seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbramos um horizonte superior, eivado pela acendra da confiança no mérito e ética aqui presente.

Ao professor Wander, pela orientação, apoio, confiança e experiências compartilhadas.

Ao nosso orientador Júlio Cesar da Silva Costa, pelo suporte no pouco tempo que lhe coube, pela suas correções e incentivos.

Agradeço aos nossos familiares, que nos deram apoio, incentivo nas horas difíceis, nos momentos de desânimo, cansaço. Pelo seus conselhos que auxiliaram e foram fundamentais para alcançar o esperado momento.

E todos que diretamente ou indiretamente fizeram parte da nossa formação, o nosso muito obrigado.

"A verdadeira motivação vem de realização, desenvolvimento pessoal, satisfação no trabalho e reconhecimento."

Frederick Herzberg.

RESUMO

O presente trabalho de conclusão de curso, teve como finalidade desenvolver um agente para reconhecer pessoas como também diferenciar uma pessoa com vida ou não através da radiação térmica emitida pelo corpo humano, com o intuito de auxiliar nas buscas e resgates em locais hostis em meio a desastres naturais.

O algoritmo é composto por uma rede neural convolucional e uma rede *Multi Layer Perceptron*(MLP), as quais possuem estruturas inspiradas em neurônios biológicos, que devido à sua capacidade de processar dados complexos e constantes o aprendizado é amplamente utilizado para resolver problemas. A primeira rede utiliza técnicas de reconhecimento por região dividindo a imagem prevendo caixas delimitadoras e ponderando-as usando probabilidade para o reconhecimento de pessoas, já a segunda rede utilizada trabalha como o modelo MLP, semelhante ao *perceptron*, mas trabalha com mais uma camada de neurônios (oculta), essa rede deverá ser capaz de identificar se a pessoa encontrada está ou não com vida.

Para objetivos secundários o algoritmo retornará um relatório da quantidade de vítimas encontradas e se estão ou não com vida. Para que esses resultados sejam alcançados, são utilizados bancos de dados que utilizam entradas e saídas conhecidas de imagens térmicas(termogramas), ou seja, aprendizado supervisionado. Os testes realizados foram satisfatórios cumprindo os objetivos desse trabalho.

Palavras-Chave: Termografia, Redes Neurais Artificiais, Espectro Eletromagnético, Histograma, Processamento de Imagem e Termograma.

ABSTRACT

The present course completion work was managed by an agent to recognize people, as well as differentiate a person living or not through damage caused by damage caused by human heat, in order to assist in barriers and recover in hostile locations amid natural disasters.

The algorithm is composed of a convolutional neural network and a Multi Layer Perceptron (MLP) network, such as which are the substances used inspired by biological neurons, which are caused by their ability to test data caused by chemical and chemical data. frequently used to solve problems. The first exchange uses region recognition techniques, dividing the image preceding bounding boxes and weighting it using the probability of people recognition, the second recovers uses works like the MLP model, looks like perceptron, but works with one more neurological disease. (hidden), this network may be able to identify whether a found person is alive or not.

For secondary objectives or returned algorithms a report of the number of victims identified and are alive or not. For what results are achieved, databases that use data and thermal image inputs (thermograms), ie supervised learning. The tests performed were satisfactory fulfilling the objectives of this work.

Keywords: Thermography, Artificial Neural Networks, Electromagnetic Spectrum, Histogram, Image Processing, Thermogram.

LISTA DE ILUSTRAÇÕES

Figura 1 – Variação de temperatura do corpo humano.	20
Figura 2 – Demonstra uma imagem do espectro do campo de visão humano.	22
Figura 3 – Visão entre os paradigmas Computação Gráfica, Processamento de Imagem e Máquina de Aprendizagem.	23
Figura 4 – Exemplo de fluxo para processamento de imagem.	23
Figura 5 – Tabela de frequência de ondas eletromagnéticas.	29
Figura 6 – Objetos sendo reconhecido por uma aplicação da visão computacional.	30
Figura 7 – Fases de um sistema de visão computacional.	31
Figura 8 – Ciclo de um processo visão computacional.	32
Figura 9 – Histograma em forma de barras.	33
Figura 10 – Histograma de uma imagem com baixo contraste.	33
Figura 11 – Representação de um neurônio biológico.	34
Figura 12 – Representação de um neurônio matemático.	35
Figura 13 – Representação do modelo MLP.	36
Figura 14 – Representação da função TanH.	37
Figura 15 – Representação da função ReLU.	38
Figura 16 – Representação da função Leaky ReLU.	39
Figura 17 – Representação da função Sigmoidal.	40
Figura 18 – Representação de uma rede convolucional.	41
Figura 19 – Fluxo de aprendizado do algoritmo <i>propagation</i>	42
Figura 20 – Processo para reconhecimento de objetos.	44
Figura 21 – Modelo de câmera de menor valor.	46
Figura 22 – Modelo de câmera de valor médio.	46
Figura 23 – Modelo de câmera de maior valor.	46
Figura 24 – Diagrama de Atividades.	48
Figura 25 – Diagrama Caso de Uso.	49
Figura 26 – Cronograma de Atividade durante o ano de 2019(Tabela).	49
Figura 27 – Instalação do Python 3.5.0 (32-bit).	50
Figura 28 – Instalação do MATLAB.	51
Figura 29 – Instalação da biblioteca OpenCV.	51

Figura 30 – Instalação da biblioteca Numpy.	52
Figura 31 – Código para extrair o histograma da imagem em python.	53
Figura 32 – Representação das imagens, cada linha é uma imagem.	53
Figura 33 – Representação das imagens de forma transposta, cada coluna é uma imagem.	54
Figura 34 – Histograma de imagem com pessoa morta.	54
Figura 35 – Histograma de imagem com pessoa viva.	55
Figura 36 – <i>Workspace</i> - Entrada de 52 imagens com 256 características em cada uma. . .	55
Figura 37 – <i>Workspace</i> - Arquivo de saída para as 52 imagens gerando 2 saídas possíveis. .	56
Figura 38 – <i>NFTool</i> - Inserindo comando.	56
Figura 39 – <i>NFTool</i> - Introdução do processo.	57
Figura 40 – <i>NFTool</i> - Selecionando Dados.	57
Figura 41 – <i>NFTool</i> - Quantidade de neurônios inseridos para teste.	58
Figura 42 – <i>NFTool</i> - Parâmetros de treinamento.	58
Figura 43 – <i>NFTool</i> - Etapa de treinamento.	59
Figura 44 – Primeiro teste de RNA com 5 neurônios.	59
Figura 45 – Curvas de aprendizado do primeiro teste de RNA.	60
Figura 46 – Curvas de aprendizado do segundo teste de RNA.	61
Figura 47 – Primeiro teste de RNA com 25 neurônios.	61
Figura 48 – Curvas de aprendizado do terceiro teste de RNA.	62
Figura 49 – Tela de entrada dos dados para treinamento.	63
Figura 50 – Primeiro teste de RNA com 25 neurônios.	63
Figura 51 – Curvas de aprendizado do quarto teste de RNA.	64
Figura 52 – Comparação entre os treinamentos realizados.	64
Figura 53 – <i>Workspace MatLab</i>	65
Figura 54 – Ambiente de testes do <i>nntool</i>	66
Figura 55 – Seleção na ferramenta <i>nntool</i>	66
Figura 56 – Resultados obtidos da RNA.	67
Figura 57 – Comandos para extrair dados da RNA treinada.	67
Figura 58 – <i>Workspace matlab</i> com os dados extraídos da rede.	68
Figura 59 – <i>Workspace Algoritimo</i>	68
Figura 60 – Importando Bibliotecas no Pyhton.	69
Figura 61 – Carregando RNA por meio do YOLO dentro do Python.	69
Figura 62 – Carregando Imagens.	70

Figura 63 – Redimensionando a imagem.	70
Figura 64 – Classificação dos dados	71
Figura 65 – Exemplificando frame a ser trabalhado na segunda RNA.	71
Figura 66 – Chamada da função da segunda RNA.	72
Figura 67 – Função de classificação do retorno da segunda RNA.	72
Figura 68 – Classe Network	73
Figura 69 – Multiplicação da entrada vs pesos.	73
Figura 70 – Tangente Hiperbólica	74
Figura 71 – Função de Ativação	74
Figura 72 – Retorno da Classificação	74
Figura 73 – Retorno da Classificação exibidos na Imagem	75
Figura 74 – Resultados da classificação referentes a figura 73	75
Figura 75 – Retorno da Classificação exibidos na Imagem	76
Figura 76 – Retorno da Classificação exibidos na Imagem	76

LISTA DE ABREVIATURAS E SIGLAS

°C	Celsius.
CV	<i>Computer Vision</i> (Visão Computacional).
CWI	<i>Centrum voor Wiskunde en Informatica</i> .
Dr	Doutor.
IA	Inteligência Artificial.
IR	Infrared (Infravermelho).
GPS	<i>Global Positioning System</i> (Sistema de Posicionamento Global).
MATLAB	<i>MATrix LABoratory</i> .
mAP	<i>mean Average Precision</i> .
ONU	Organizaçāo das Naçōes Unida.
OpenCV	<i>Open Source Computer Vision Library</i> .
MLP	<i>Multi Layer Perceptron</i> (Perceptron de Múltiplas Camadas).
R-CNN	<i>Regiões com redes neurais convolucionais</i> .
RNA	Rede Neural Artificial.
RP	Reconhecimento de Padrões.
SVA	Sistema de Visão Artificial.
VNT	<i>Visual Nerve Tracer</i> .

SUMÁRIO

1	INTRODUÇÃO	15
1.1	Tema	15
1.2	Problema	15
1.3	Hipótese	16
1.4	Justificativa	16
1.5	Objetivos	16
1.5.1	Objetivo Geral	16
1.5.2	Objetivos Específicos	16
2	METODOLOGIA	17
2.1	Tipo de Pesquisa	17
2.2	Método de Realização	17
2.3	Critérios de Delimitação	17
3	REFERENCIAL TEÓRICO	18
3.1	Termografia	18
3.1.1	Introdução	18
3.1.2	Percepção de Calor	18
3.1.2.1	Dispositivo Termográfico	19
3.1.3	Temperatura Corpo Humano	20
3.2	Imagem	21
3.2.1	Introdução	21
3.2.2	Definindo Imagem Digital	21
3.2.3	Definindo Imagem Infravermelha	21
3.2.3.1	Paleta	22
3.2.3.2	Percepção das Cores	22
3.2.4	Processamento de Imagem	22
3.2.4.1	Aquisição de Imagem	24
3.2.4.2	Pré-processamento	24
3.2.4.3	Segmentação	25
3.2.4.4	Representação e Descrição	25
3.2.4.5	Reconhecimento	26

3.2.4.6	Interpretação	26
3.3	Câmera	27
3.3.1	Introdução	27
3.3.2	Sensores	27
3.3.3	Espectro Eletromagnético	28
3.4	Visão Computacional	30
3.4.1	Introdução	30
3.4.2	Técnicas de Visão Computacional	31
3.4.3	Histograma	33
3.5	Rede Neural Artificial	34
3.5.1	Introdução	34
3.5.2	Neurônio Biológico	34
3.5.3	Neurônio Matemático	34
3.5.4	Funções de Ativação	36
3.5.4.1	Função de Etapa Binária	36
3.5.4.2	Função Linear	37
3.5.4.3	Função TanH	37
3.5.4.4	Função ReLU	38
3.5.4.5	Função Leaky ReLU	38
3.5.4.6	Softmax	39
3.5.4.7	Função Sigmoidal	39
3.5.5	Redes Neurais Convolucionais	40
3.5.5.1	Extração de características	40
3.5.5.2	Mapeamento de características	41
3.5.5.3	Subamostragem	41
3.5.6	Backpropagation	41
3.6	Ferramentas	42
3.6.1	Introdução	42
3.6.2	OpenCV	42
3.6.3	Python	43
3.6.4	MATLAB	43
3.6.5	YOLO	43
4	DESENVOLVIMENTO	45

4.1	Delimitação do Problema	45
4.1.0.1	Visão Inicial	45
4.2	Engenharia de Software	46
4.2.1	Introdução	46
4.2.2	Gerência de Escopo	47
4.2.2.1	Requisitos Funcionais	47
4.2.2.2	Requisitos Não Funcionais	47
4.2.2.3	Diagrama de Atividades	48
4.2.2.4	Diagrama Caso de Uso	48
4.2.3	Gerência do Tempo	49
4.3	Configuração de Ambientes	50
4.3.1	Introdução	50
4.3.2	Instalando Python	50
4.3.3	Instalando MATLAB	50
4.3.4	Bibliotecas	51
4.3.4.1	Instalando OpenCV	51
4.3.4.2	Instalando Numpy	52
4.4	Roteiros de Treinamentos RNA	52
4.4.1	Introdução	52
4.4.2	Preparando a base de dados para treino	52
4.4.3	Criando RNA no MATLAB	55
4.4.4	Primeiro Treinamento	59
4.4.5	Segundo Treinamento	60
4.4.6	Terceiro Treinamento	61
4.4.7	Quarto Treinamento	62
4.4.8	Comparação dos treinamentos	64
4.4.9	Testando a RNA no MATLAB	65
4.4.10	Salvando RNA	67
4.5	Codificação	68
4.5.1	Introdução	68
4.5.2	Obtendo a RNA	68
4.5.3	Separando objetos a serem trabalhados	71
4.5.4	Utilizando a rede gerada no MatLab	72

5	CONSIDERAÇÕES FINAIS	77
5.1	Trabalhos futuros	77
	REFERÊNCIAS	78
	ANEXO A – CÓDIGO PYTHON RNA YOLO	83
	ANEXO B – CÓDIGO PYTHON RNA MLP	86

1 INTRODUÇÃO

O presente trabalho de conclusão de curso, tem como princípio auxiliar na busca e no resgate de pessoas com o uso aplicado da Inteligência artificial e visão computacional. Devido aos recentes casos de desastres no Brasil, como o rompimento da barragem em Brumadinho, em janeiro de 2019, a tragédia em Mariana, em 2015 e as enchentes e deslizamentos na Região Serrana do Rio de Janeiro, janeiro de 2011, onde acompanhado por reportagens, notícias de televisão e internet, as incansáveis buscas de sobreviventes.(OLIVER, 2015)

A busca de vítimas, em desastres se torna muito cansativa e exaustiva em muitas das vezes, e para que o maior número de vítimas dos desastres sejam encontradas propôs-se o desenvolvimento de uma rede neural que utilizando a visão computacional em imagens termográficas (termogramas) seja capaz de identificar pessoas exibindo o número de vítimas encontradas.

O algoritmo a ser desenvolvido deve tratar a imagem termográfica extraíndo suas características para ser realizada a classificação. O algoritmo de classificação trabalha com duas redes neurais sendo a primeira utilizando o YOLOv3 que terá o trabalho de reconhecer a vítima e conversar com a segunda rede que trabalha com um modelo MLP com mais de uma camada de neurônios que terá o trabalho de identificar o estado de vida. Utilizando o MATLAB para criar a segunda rede, e extraír seus pesos, onde será utilizado na linguagem em Python junto com bibliotecas do openCV e numpy.

1.1 Tema

Agente para detecção de pessoas utilizando técnicas de Inteligência Artificial (IA) através de termografia para tomada de decisão.

1.2 Problema

Seria possível o desenvolvimento de um agente com Visão Computacional (CV) para auxiliar na busca e identificação de vítimas em missões de resgate em locais de difícil acesso?

1.3 Hipótese

A frequência de acidentes por desastres naturais em escala global e nacional tem aumentado nos últimos anos, com o uso da IA aplicando técnicas de CV, espera-se que um agente possa auxiliar na busca e identificação de vítimas em missões de resgate (PIERRO, 2018 *apud* CATARINA, 2013). Portanto, com os recentes avanços da IA e na evolução tecnológica das câmeras o presente trabalho propõe o desenvolvimento de um agente que auxilie na busca e identificação de vítimas com ou sem vida.

1.4 Justificativa

Segundo informações da Organização das Nações Unidas (ONU) o Brasil é o único país das Américas que está na lista dos top 10 países com maior número de pessoas afetadas por desastres naturais entre os anos de 1995 a 2015. "Em média, 335 desastres naturais foram registrados anualmente entre 2005 e 2014, um aumento de 14% entre o período de 1995 – 2004 e quase o dobro da média de 1985 – 1995"(BRASIL, 2015).

Com o intuito de resgatar o máximo de pessoas com vida nos locais afetados pelos desastres, o presente trabalho demonstra o resultado da pesquisa de técnicas de IA com uso da termografia que resultou no desenvolvimento de um agente de *software* com o propósito de auxiliar a equipe de resgate na localização de vítimas em locais de difícil acesso com maior precisão.

1.5 Objetivos

1.5.1 Objetivo Geral

Desenvolver um protótipo de um agente capaz de auxiliar na busca e na identificação possíveis vítimas em locais de difícil acesso.

1.5.2 Objetivos Específicos

- Analisar fotos retiradas por câmera termográfica e identificar possíveis vítimas;
- Classificar as vítimas indicando se estão ou não com vida;
- Apresentar relatório para equipe de resgate para tomada de decisão.

2 METODOLOGIA

A elaboração deste trabalho, se baseou na pesquisa exploratória, proporcionando maior familiaridade e tornando mais claro ou levantar hipóteses do problema apresentado.

2.1 Tipo de Pesquisa

O presente trabalho possui caráter bibliográfico e documental, pois para sua fundamentação foram investigados artigos, livros, revistas redes eletrônicas dos principais conceitos e práticas associados ao tema.

2.2 Método de Realização

Sendo assim, o primeiro passo foi a realização de uma pesquisa dos números de acidentes por desastres naturais no Brasil constatando o fato de que grande número de pessoas são afetadas. Em seguida realizamos uma abordagem técnica observatória, onde o observador tem de registrar a maior quantidade de informação possível e captar uma perspectiva ‘de dentro’.

2.3 Critérios de Delimitação

Observado o comportamento e ações dos socorristas em desastres naturais, analisamos a dificuldade dos socorristas para encontrar as vítimas devido a dificuldade encontradas em locais hostis muitas das vezes indo para o lado oposto da vítima que é encontrada depois de incansáveis buscas.

Através deste trabalho busca-se auxiliar e direcionar os socorristas na direção correta identificando onde as vítimas se encontram, para que recebam atendimento de imediato.

3 REFERENCIAL TEÓRICO

3.1 Termografia

3.1.1 Introdução

Neste capítulo descreve o nascimento de termografia desde a percepção de calor a primeira imagem térmica.

3.1.2 Percepção de Calor

O pontapé inicial se deu nos tempos mais remotos da história, onde os filósofos e médicos gregos (Platão, Aristóteles, Galeno e Hipócrates) distinguiram e se fascinaram com a semelhança entre o calor e a vida, onde não se questionava a origem do calor humano, mas especulavam-se as formas pelas quais o calor se dispersava do corpo e tendo em vista a respiração como mecanismo evidente de resfriamento, pois sentia-se a temperatura mais quente do que o ar ao ser exalado (BRIOSCHI, 2017 *apud* LOMAX, 1979).

Hipócrates notou temperaturas variadas em diferentes partes do corpo humano e considerou o aumento de calor inato do corpo humano como principal diagnóstico de doenças. “[...] quando uma parte do corpo é mais quente ou mais fria do que o restante, então a doença está presente nesta parte” (BRIOSCHI, 2017 *apud* ADAMS, 1939).

Hipócrates passou a sentir o calor com o dorso da sua mão e então confirmava esfregando a área com lama observando onde secava e endurecia primeiro. Nasceu então a termografia (BRIOSCHI, 2017 *apud* ADAMS, 1939).

Em 1592 o astrônomo Galileu retomou os estudos de calor corporal pela descoberta e desenvolvimento do primeiro termômetro a ar. Mais tarde, Sanctorius, modificou o termômetro, dividiu o seu próprio e descreveu-o em grandes detalhes. A partir desse momento outros estudiosos como Boullian que em 1659, modificou o termômetro de Sanctorius introduzindo mercúrio dentro de um tubo de vidro; Fahrenheit, Celsius e Joule contribuíram com o desenvolvimento das escalas termométricas. Então surgiu um século após Spurgin, em 1857, construiu um termoscópio que comparando a temperatura superficial, foi capaz de diagnosticar tumores de mama, distinguindo que o calor do tumor era vários graus maiores que do tecido periférico (BRIOSCHI, 2017 *apud* COHEN, 1967).

A Termologia Quioprática que em um empenho ao utilizar o diferencial de temperatura como instrumento de análise, D.D. Palmer utilizou a superfície sensitiva do dorso da mão para localizar áreas hipertérmicas ao longo da coluna espinhal pensando-se que estas áreas de concernente aumento de temperatura ao redor dos tecidos poderiam identificar nervos inflamados devida compressão ou subluxação (BRIOSCHI, 2017 *apud* DYE, 1939).

3.1.2.1 Dispositivo Termográfico

No início do século 20 Doss Evins, engenheiro elétrico, estudante de B.J. Palmer, criou um dispositivo sensível a calor no qual localizava semi quantitativamente áreas para espinhais de aumento de calor (BRIOSCHI, 2017 *apud* DYE, 1939).

A partir desse momento foram produzidos diversos equipamentos para diagnosticar áreas com alto calor, então no início dos anos 60 surgiu a Termografia médica moderna onde Adelman projetou o Visual Nerve Tracer(VNT) para realizar a mesma análise espinhal baseado em distintos princípios fisiológicos (BRIOSCHI, 2017 *apud* NOVICK, 1969).

O VNT foi um medidor de reflexão fotoelétrica; seus dois probes paraespinhais emitem luz visível, filtrava para a passagem de ondas de cerca de 556 nm apenas de comprimento. Essa onda era fortemente absorvida pela hemoglobina presente nas hemácias (vermelho). A luz refletida era, então detectada por uma célula fotoelétrica de cada probe, e os probes contralaterais eram conferidos eletronicamente para produzirem uma linha num gráfico mostrando de flexões em áreas com fluxo sanguíneo cutâneo adulterado (BRIOSCHI, 2017 *apud* NOVICK, 1969).

No final do século XVI Della Porta, começou o estudo e experimentos da Radiação Infravermelha (IR). William Herschel, dois séculos mais tarde utilizando um espectroscópio, descobriu que o sol emitia raios infravermelhos. Esta descoberta e sua reação com a luz não se tornaram claras até metade do século XIX, quando seu filho, Sir John Herschel, produziu em papel a primeira imagem térmica, sendo um pioneiro no campo da fotografia (BRIOSCHI, 2017 *apud* PUTLEY, 1982).

Langley na mesma época desenvolver o bolômetro, aparelho capaz de captar o calor de objetos vivos a uma distância superior a 400m (BRIOSCHI, 2017 *apud* RASK, 1979).

A tecnologia IR alavancou após a Segunda Guerra Mundial, mais era restrita ao uso militar. Czerny adicionando novos termistores que eram acoplados em um aparelho de detecção de imagem teve o resultado de um novo instrumento que detectava, embora rudimentar, movimentos de tropas em campos em terrenos e movimentos de navios à noite. Anos depois

o Dr. Ray Lawson, usou o equipamento para medicina experimental (BRIOSCHI, 2017 *apud* CURCIO; HABERMAN, 1971).

O equipamento consistia em um bolômetro termistor que captava o calor emitido e transformado em sinais elétricos. Os sinais iluminavam um tubo de gás que brilhava com uma intensidade proporcional à radiação detectada pelo termistor. A luz era então refletida em um filme fotográfico e produzia um termograma (são as imagens que foram convertidas pelas câmaras termográficas) (BRIOSCHI, 2017 *apud* CURCIO; HABERMAN, 1971).

No final dos anos 60 a empresa AGA produziu o AGA Thermovision, cuja habilidade de gerar uma imagem semelhante TV, permitindo observar as imagens instantaneamente do corpo humano (BRIOSCHI, 2017 *apud* RYAN, 1969).

Apesar deste pobre presságio para uso médico, sofisticados equipamentos eletrônicos IR foram inseridos no início dos anos 70 por várias industriais. Um importante avanço foi o desenvolvimento de uma modalidade de isotermia colorida (BRIOSCHI, 2017 *apud* RYAN, 1969).

3.1.3 Temperatura Corpo Humano

A emissão de calor do corpo humano em condições normais de operação é de 36,5 a 37°C podendo ter variações de acordo com a situação em que se encontra podendo gerar hipertermia e hipotermia, sendo a hipertermia a elevação da temperatura do corpo relacionada à incapacidade do corpo de promover a perda de calor para o ambiente em que se encontra ou, ainda, reduzir a produção de calor, e a hipotermia é a quando a temperatura central do corpo humano cai abaixo de 35°C (EDUCAÇÃO, 2018). A figura 1 demonstra a variação de temperatura do corpo humano.

Figura 1 – Variação de temperatura do corpo humano.

Variação de Temperatura do Corpo	
Estado Térmico	Temperatura (°C)
Sub-normal	34-36
Normal	36-37
Estado febril	37-38
Febre	38-39
Febre alta (pirexia)	39-40
Febre muito alta (hiperpirexia)	40-41

Fonte – EDUCAÇÃO, 2018.

Com objetivo de identificar a probabilidade de ser um ser humano vivo, trabalhamos com a temperatura de 35 a 37 °C (EDUCAÇÃO, 2018).

3.2 Imagem

3.2.1 Introdução

O capítulo seguinte será apresentado de forma objetiva a definição, tipos e será descrito o processo de manipulação de uma imagem digital e imagem infravermelha.

3.2.2 Definindo Imagem Digital

Conforme dito por (GONZALEZ; WOODS, 1992), a definição de imagem pode ser entendida como uma função bidimensional $f(x,y)$, em que as variáveis x e y representam as coordenadas espaciais. A distância entre qualquer este par variável (amplitude) define a intensidade ou nível de cinza presente na imagem.

Uma forma de representar uma imagem e interpretá-la como uma coleção de pontos, cada um deles chamado de pixel, sinônimo para *Picture Element (Elemento de Imagem)*. A aparência de cada pixel é então, codificada, e a imagem completa é representada como uma coleção de tais *pixels* codificados. Tal coleção é chamada de mapa de bits. Essa abordagem é popular porque muitos dispositivos de visualização, como impressoras e monitores, operam usando o conceito de pixels. Assim, as imagens na forma de mapas de bits são facilmente formatadas para visualização (BROOKSSHEAR; SMITH, 2013).

3.2.3 Definindo Imagem Infravermelha

Segundo Gonzales e Woods (2009), o sistema de imagem infravermelha trabalha na banda 10,0 a 13,4 em relação aos comprimentos de ondas, e tem a capacidade única de observar fontes fracas de emissões visíveis de infravermelhos próximos presentes na superfície da terra. Gonzales e Woods (2009) relata que, o processo de formação de imagens infravermelhas é feito a união entre a banda infravermelha e a banda visível para obtenção e aplicação de imagem em áreas, como sensoriamento remoto, policiamento, microscopia ótica e astronomia. Este processo que utiliza bandas visíveis é relevante o uso de inspeção visual para criar produtos manufaturados concebidos por processamentos de imagens. Estes produtos gerados nos processos de inspeção, são fragmentos de um todo para que poderá ser analisado em determinada área ou contexto aplicado.

3.2.3.1 Paleta

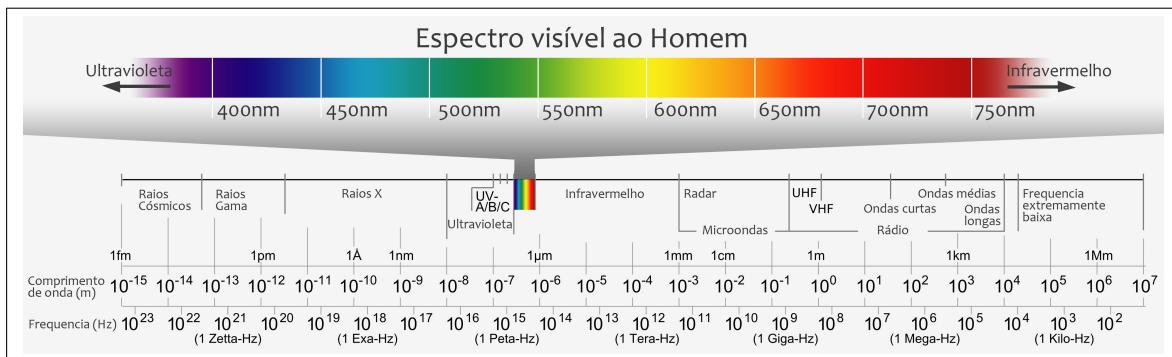
Uma paleta serve para representar as medidas de radiação em cada ponto do termograma sendo representada por cores específicas, sendo mais objetivo na identificação de padrões e relacionamento com a imagem (CARVALHO, 2012-2013).

3.2.3.2 Percepção das Cores

O ser humano é capaz de perceber apenas o espectro de luz visível. Computadores reconhecem cores de forma consistente e precisa fazendo o uso de técnicas de visão computacional. Uma determinada cor pode ser compreendida de forma única por indivíduo, ou seja, depende do observador e da cor dos objetos próximos Conci, Azevedo e Leta (2009).

De maneira geral, nossa retina pode captar, com diferentes sensibilidades, comprimentos de onda entre 400 e 700 nanômetros e somos incapazes de visualizarmos a radiação IF a olho nu sem ajuda de equipamentos. Utilizamos os equipamentos de IF para gerar imagens térmicas do que não conseguimos enxergar. Seguindo nesse pensamento foi iniciado o estudo em Termografia em diversas áreas (medicina, militar, elétrica, etc.) (LACERDA; RESENDE, 2014). A figura 2 demonstra uma imagem do espectro do campo de visão humano.

Figura 2 – Demonstra uma imagem do espectro do campo de visão humano.

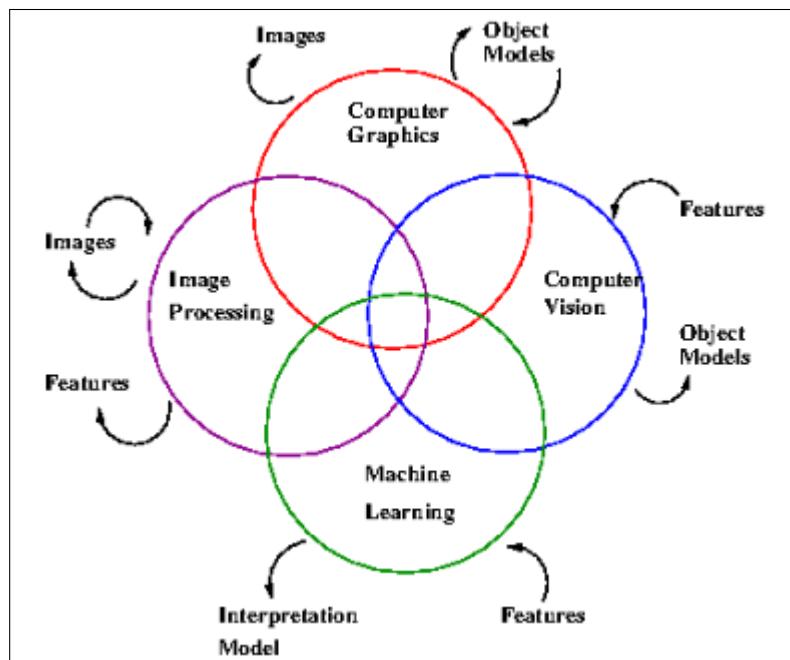


Fonte – LÓPEZ, 2016.

3.2.4 Processamento de Imagem

Segundo Gonzalez e Woods (1992), processamento de imagem é motivo de debate entre autores pois, alguns autores defendem visão de que é um subgrupo de estudo pertencente a visão computacional porém, há quem defenda que seja um dos alicerces para o processo aprendizado de máquinas presente na IA. A figura 3 traz um modelo de visão deste paradigma.

Figura 3 – Visão entre os paradigmas Computação Gráfica, Processamento de Imagem e Máquina de Aprendizagem.

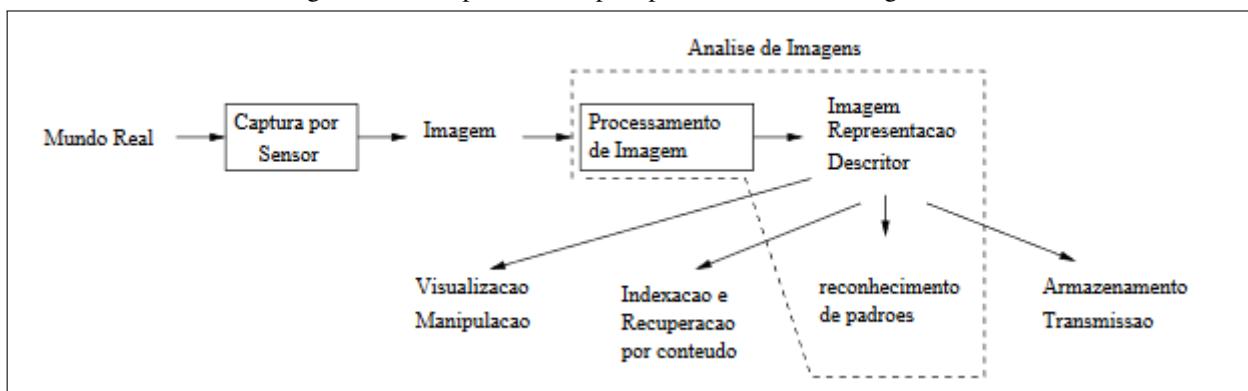


Fonte – FALCÃO, 2014.

Conforme dito por Gonzalez e Woods (1992), o paradigma pode ser definido como uma área de estudo na qual tem como entrada uma imagem qualquer e gera em sua saída uma imagem para diferentes alvos conforme a demanda. Mediante a este pensamento, o processamento de imagem não implica uma simples análise para obter como saída um cálculo de intensidade média em uma imagem, ou seja, ter uma única saída.

O forte deste estudo é ter uma imagem base como entrada para gerar parâmetro para novos dados como Aprendizagem de Máquina ou Reconhecimento de Padrão. Portanto, o objetivo é simular uma visão humana inteligente através do poder computacional desenvolvido por meio de algoritmo de IA. Visão na qual seja capaz de aprender e diferenciar informações visuais do mundo real (GONZALEZ; WOODS, 1992).

Figura 4 – Exemplo de fluxo para processamento de imagem.



Fonte – FALCÃO, 2014.

A figura 4 traz um fluxo representando como o processo acontece em um Sistema de Visão Artificial (SVA) que faz uso de processamento de imagem e armazenamento (FILHO; NETO, 1999).

Pode-se compreender e distinguir que o nível que se encontra o processamento de imagem e análise de imagem. Um reconhecimento de objetos, lugares ou padrões só acontece após processar e analisar a matriz de dados gerada no processo. O conceito pode ser interpretado e subdividido em três níveis (GONZALEZ; WOODS, 1992):

- Baixo: Pré-processamento;
- Médio: Segmentação, Representação, Descrição e Reconhecimento;
- Alto: Interpretação.

3.2.4.1 Aquisição de Imagem

Uma imagem pode ser capturada de um ambiente por meio de diferentes aparelhos como *smartphones*, câmeras de fotográficas, câmeras de vídeo, scanners ou satélites. Compreende-se que a aquisição de imagem está associado a combinação de uma fonte de "iluminação" e o reflexo ou absorção da energia fornecida pelos elementos da "cena"(GONZALEZ; WOODS, 1992).

"A energia da radiação eletromagnética conduz de forma analógica a informação sobre os objetos e no sensor um conversor analógico/digital converte essa informação em um valor digital, codificado por unidade chamada de pixel"(MENESES *et al.*, 2012).

Cada tipo de aparelho realiza este processo de captura conforme a sua finalidade e necessidade (GONZALEZ; WOODS, 1992). O aparelho é sensível a uma faixa de energia no espectro eletromagnético, que produz na saída de um sinal elétrico proporcional ao nível de energia detectado. Em seguida será empregado o digitalizador, com a função de converter este sinal elétrico analógico em uma informação digital (FILHO; NETO, 1999).

3.2.4.2 Pré-processamento

Será feito o pré-processamento obtido na saída do processo de aquisição de imagem. O objetivo é retirar toda obstrução produzida pela radiação emitida pelos sensores na fase de aquisição de imagem. Aprimorar a estrutura da imagem, ou seja, dar qualidade a imagem em aspecto de contraste e nitidez (GONZALEZ; WOODS, 1992).

Caracteriza-se por um processo de baixo nível porque trabalha diretamente com os valores de intensidade de cada pixel reduzindo as imperfeições como ruídos, contraste inadequado, brilho inadequado, dado interrompido ou conectado de forma errônea. A imagem resultante será digitalizada com qualidades aperfeiçoadas em relação a original (FILHO; NETO, 1999).

3.2.4.3 Segmentação

A segmentação tem a tarefa de atribuir rótulos semânticos de cada pixel da imagem em um como “carro”, “casa”, “cachorro”. Para a atribuição dos rótulos é necessário uma precisão e localização de entidades visuais, como classificação no nível da imagem para identificar os contornos dos objetos ou detecção no nível de caixa delimitadora com técnicas de descontinuidade que subdivide a imagem com base nas alterações bruscas de intensidade e limiarização que se baseia na diferença dos níveis de cinza que compõem diferentes objetos de uma imagem (CHEN *et al.*, 2014).

3.2.4.4 Representação e Descrição

(GONZALEZ; WOODS, 1992) afirma que a saída da fase de segmentação é dada por dados brutos de pixel, gerando a necessidade de convertê-la em uma forma oportuna para o processamento do computador podendo ser utilizado dois tipos de representação sendo que em algumas aplicações elas coincidem:

- Representação Limite: usada quando o objetivo está nas características da forma obtida, como por exemplo bordas;
- Representação Regional: usada quando o objetivo está na peculiaridade reflectiva, como textura e cor;

Definir a representação a ser usada é o primeiro passo para transformar os dados brutos em uma forma oportuna para o processamento. Também deve ser utilizado um procedimento para rotular as características de interesse sendo realçadas, possibilitando a extração das características resultando na diferenciação de uma classe de objetos de outra (GONZALEZ; WOODS, 1992).

3.2.4.5 Reconhecimento

De acordo com SOUZA (1999), na fase de reconhecimento o objetivo é identificar o que cada uma das classes representa com um respectivo valor.

O reconhecimento de padrões é grande devido ao fato de que na vida humana tudo se dá em formas e padrões. Desde a fala, escrita e a compreensão das formas tudo envolvendo padrões. Identificar padrões é uma tarefa extremamente complexa, onde se avalia as situações em termos de padrões e da condição que as constituem, descobrir relações existentes no meio para decifrá-la e aprender adaptando-se (SOUZA, 1999).

Há muitas técnicas de Reconhecimento de Padrões (RP) no qual é dividida em Estatística, Estrutural e Neural. Sendo a Estatística um conjunto de vetores ou n-tuplas com medidas, características e métodos extraídos da imagem sendo utilizados para separar as classes, a Estrutural representada por padrões como árvores e strings e métodos de reconhecimento baseados em casamento de modelos e símbolos que tratam como sentenças, a partir de uma linguagem artificial e a Neural realiza o reconhecimento utilizando Redes Neurais Artificiais. Sendo considerado como um tipo particular de reconhecimento estatístico sendo suas características na forma de vetores e n-tuplas por alguns autores (BISHOP, 1995).

3.2.4.6 Interpretação

A interpretação de imagens é uma tarefa complexa tendo grandes dificuldades pela quantidade de dados a serem analisados e pela falta de ferramentas de processamento para receber os dados iniciais para gerar os resultados desejados, sendo forçados a considerar aproximações com razoáveis chances de sucesso, considerando duas restrições limitando a generalidade do problema e a incorporação do reconhecimento humano no processo (GONZALEZ; WOODS, 1992).

Utilizando os três formalismos principais para representar o conhecimento sendo elas lógica formal onde o conhecimento pode ser aberto através de regras lógicas, redes semânticas que através de grafos demonstra o relacionamento entre elementos de uma imagem, e sistema de produção ou regra base é utilizado numa larga escala de aplicativo em processamento de imagem digital (GONZALEZ; WOODS, 1992).

3.3 Câmera

3.3.1 Introdução

O mundo real para o ser humano são facilmente sentidos, interpretados e compreendidos para gerar conhecimentos, sempre através dos órgãos de sentido (visão, audição, tato, olfato e paladar). São eles que norteiam o ser humano para captar informações como cor, frio, gostos, calor, energia e outras sensações. No mundo das máquinas não é muito diferente, pois, os órgãos de sentido são os sensores, ou seja, dispositivos incorporados em placas prontas para interpretar um ambiente. Estes dispositivos são capazes de capturar dados e entregar aos computadores para solucionar um determinado problema, suas especificações variam de acordo com o modelo. Dentre elas o modelo da Flir possui as seguinte especificações, faixa de temperaturas Cena: -4 F a 248 F (-20 C a 120 C), temperatura de Operação: 32 F a 95 F (0 C a 35 C), peso: 78 gramas, dimensões: L = 72mm x W = 26 milímetros x H = 18 milímetros, câmera visível: VGA (usado para FLIR® MSX® blending) e sensibilidade: capacidade de detectar diferenças de temperatura tão pequenas quanto 0,18 F (0,1 C) (FLIR, 2019a).

Câmeras térmicas, por exemplo, são usados sensores também uma para cada situação. O aparelho é capaz de detectar energia (calor) infravermelha e através de um sinal analógico e convertê-la em um sinal eletrônico sem um contato físico direto com o objeto. O sinal captado é processado para produzir uma imagem ou vídeo com as características térmicas, pronto para um imediato cálculo de temperatura. Este calor identificado por ela pode ser quantificado ou medido com muita precisão (FLIR, 2019a).

3.3.2 Sensores

Câmeras Infravermelhas (IV) móveis do mercado utilizam geralmente 2 modelos de sensores térmicos chamados de “Refrigerados” e “Não Refrigerados” (FLIR, 2019b).

Segundo Flir (2019b) câmeras que tem sensores de imagem integrados com um resfriador criogênico são chamadas de câmeras refrigeradas. Os resfriadores ficam selados a vácuo em um recipiente. A perda de calor oferecida pelo processo de arrefecimento é importante para que os semicondutores funcionem e façam a redução de ruídos induzido termicamente a um nível abaixo do sinal da cena que está sendo capturada. Este resfriador geralmente possui peças móveis com uma construção próxima das tolerâncias mecânicas que se desgastam, assim como o gás hélio/nitrogênio que lentamente passa pelas vedações de gás. Portanto, estes os resfria-

dores passam por reconstrução eventualmente após 10.000-13.000 horas de uso. Câmera com sensor refrigerado está intimamente associado ao conjunto de necessidade, ou seja, tem maior utilidade quando precisa de:

- Qualidade na imagem;
- Alta velocidade de resposta;
- Grau de análise elevado em questão do perfil térmico em alvos menores;
- Visualização de fenômenos térmicos em uma parte específica do espectro eletromagnético;
- Caso tenha necessidade de integração com outros dispositivos de medição.

O tipo de câmera que usa sensores ‘resfriados’ tem por natureza em ser muito custoso, tanto no aspecto da compra quanto em relação às manutenções. No entanto, possui câmeras que não utilizam sensores refrigerados, o que torna o equipamento mais acessível. Realizando constantemente comparações entre a temperatura e o funcionamento do sensor. Neste caso, o calor fornecido pela onda eletromagnética é capturado do ambiente através de oscilações de resistência, tensão ou corrente (FLIR, 2019b).

3.3.3 Espectro Eletromagnético

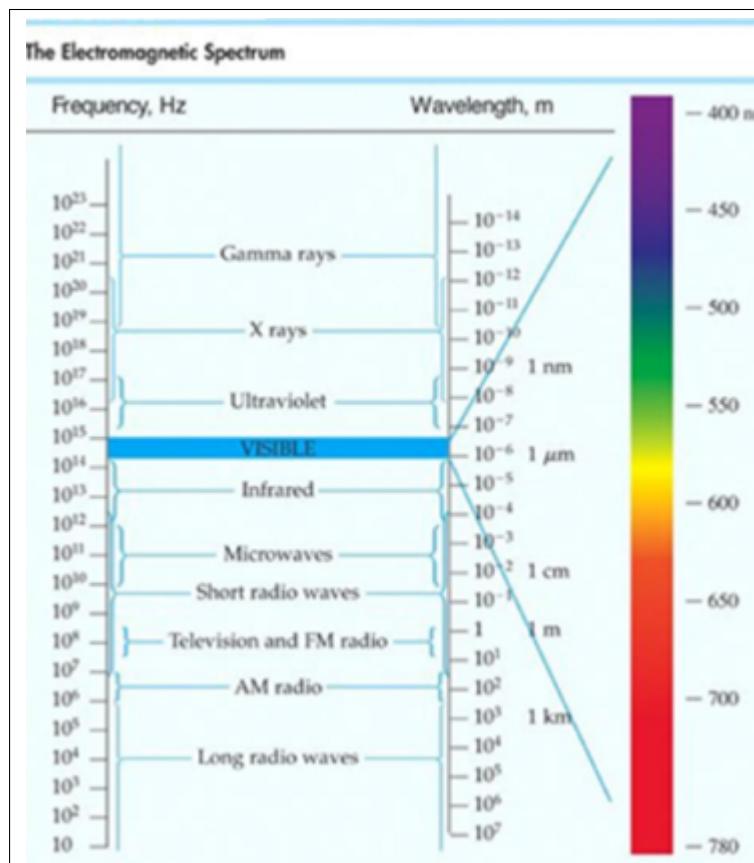
Eletricidade e magnetismo são áreas do estudo Eletrostático da Física que possuem uma íntima ligação, sendo dividido por alguns autores entre fenômenos elétricos e fenômenos magnéticos. O Eletromagnetismo é um ponto de ligação entre às duas áreas, onde estuda o relacionamento de fenômenos elétricos com o fenômeno magnético (SALMERON, 2008). A diversidade de radiações produzidas pelo sol é comumente classificada em faixas específicas de comprimentos de onda que compartilham características semelhantes de interação com a matéria ou aplicação prática na Terra. Essa classificação é denominada Espectro Eletromagnético e sua divisão é subjetiva, podendo variar dependendo da aplicação e da referência utilizada (ZANOTTA; FERREIRA; ZORTEA, 2019).

Quando um campo elétrico ou campo magnético varia em um espaço de tempo, ocorre uma indução do outro campo na região do espaço adjacente ao campo variante. Avalia-se a ocorrência de uma perturbação eletromagnética constituída por campos elétricos e magnéticos variando com o tempo e que pode se propagar de uma região do espaço para outra, e que essa

perturbação possui características de uma onda eletromagnética. Ondas eletromagnéticas cobrem um intervalo de espectro complementarmente amplo de comprimento de onda e frequência (YOUNG; FREEDMAN, 2009).

Na figura 5 demonstra todos os intervalos de frequências entre as ondas, desde as baixas frequências até a mais alta.

Figura 5 – Tabela de frequência de ondas eletromagnéticas.



Fonte – FALCÃO, 2014.

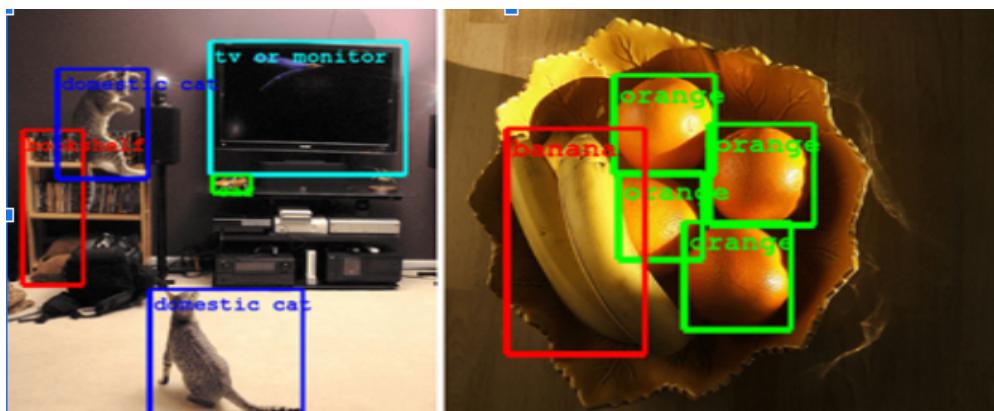
Young e Freedman (2009) cita que, os olhos humanos não conseguem enxergar todos os intervalos de ondas eletromagnéticas. Um exemplo simples de baixa frequência é a luz visível emitida por um filamento de lâmpada incandescente. Outro exemplo são as câmeras, elas possuem um dispositivo que libera um feixe de radiação infravermelha. Analisando as características da radiação infravermelha refletida do sujeito, a câmera comprehende a distância do sujeito e de forma autônoma corrige o seu foco. Assim quanto maior a onda de calor, radiação térmica ou radiação infravermelha produzida pelo objeto analisado, maior será o espectro magnético.

3.4 Visão Computacional

3.4.1 Introdução

O reconhecimento de imagens realiza a análise de pixel e padrão de imagens é uma parte integrante do processo de visão computacional que envolve tudo, desde reconhecimento de objetos e caracteres até análise de texto e expressões humanas. O reconhecimento de imagem de hoje, ainda na maior parte, apenas identifica objetos básicos, veja a representação na figura 6. Mesmo crianças podem fazer isso, mas o potencial da visão computacional é sobre-humano: ser capaz de ver claramente no escuro, através de paredes, em longas distâncias e processar todos esses dados rapidamente e em volume maciço de informações (ACADEMY, 2018).

Figura 6 – Objetos sendo reconhecido por uma aplicação da visão computacional.



Fonte – ACADEMY, 2018.

De acordo com Backes e Junior (2016) podemos entender visão computacional como a área de estudo que tenta passar para máquinas a incrível capacidade da visão. No entanto a visão consiste em captar imagens e melhorá-las.

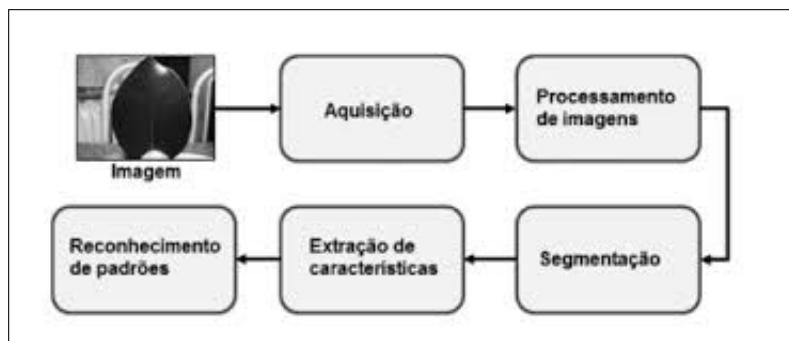
Conforme citado por (BALLARD; M., 2007) a visão computacional é a ciência que estuda e desenvolve tecnologias que permitem máquinas enxergar e extraír características do ambiente, através de imagens capturadas por câmeras de vídeo, sensores, scanners e outros dispositivos.

Segundo Backes e Junior (2016) as decisões a serem tomadas em visão computacional são baseadas no que se pode capturar de uma imagem, com o modo de percepção humana do ambiente tornando um sistema constituído de várias fases:

- Aquisição;
- Processamento de imagens;

- Segmentação;
- Extração de características;
- Reconhecimento de padrões.

Figura 7 – Fases de um sistema de visão computacional.



Fonte – BACKES; JUNIOR, 2016.

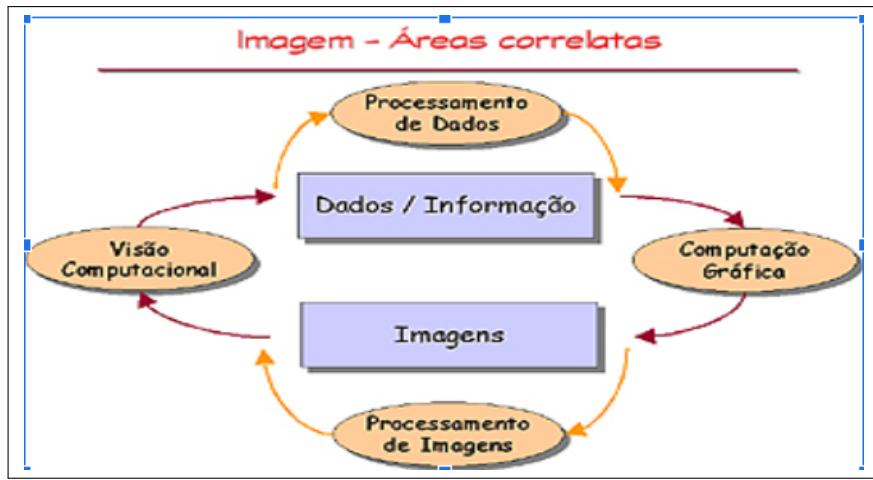
Analizando as fases da figura 7 podemos entender o processo de automatizar os serviços que o sistema visual humano exerce, assim como um médico pode reconhecer um tumor em uma tomografia ou uma pessoa poderia identificar o próprio rosto em uma fotografia, tais sistemas poderiam fazer o mesmo (Szeliski, 2010).

- Reconhecimento ótico de caracteres: leitura de códigos postais escritos a mão, tarefa essa que pode ser efetuada com o uso de uma rede.
- Reconhecimento de pessoas: em ambiente hostil onde a visão humana é limitada. O recurso de visão computacional pode ajudar a identificar vítima em situações precárias.
- Modelagem 3D: usando fotogrametria que é uma técnica utilizada para extrair medidas rigorosas para a construção de modelos em três dimensões a partir de fotografias aéreas.

3.4.2 Técnicas de Visão Computacional

Segundo Gonzalez e Woods (1992) todas as técnicas de visão computacional são equivalentes as técnicas de alto nível do processamento de imagens, como o reconhecimento e interpretação de objetos e padrões, classificação de objetos.

Figura 8 – Ciclo de um processo visão computacional.



Fonte – ALBUQUERQUE, 2001.

Filho e Neto (1999) complementam que o processamento de imagens lida com informações sobre as imagens para interpretação humana e a visão computacional é a análise automática realizada pelo computador de informações obtidas a partir de uma ou várias imagens representados na figura 8.

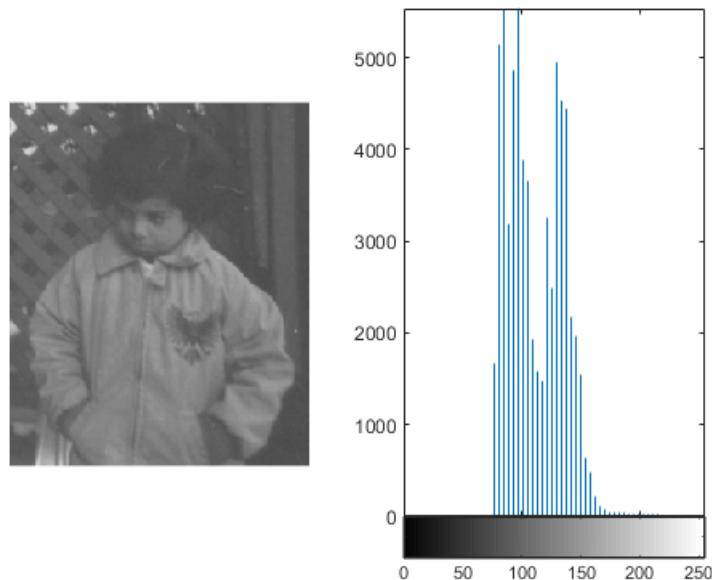
De acordo com Filho e Neto (1999) os tópicos abaixo compreendem as seguintes etapas:

- Processamento de Dados: É toda vez que partimos de uma informação, sob a forma de dados (por exemplo números em planilha eletrônica) e obtemos uma nova informação, através de um processamento automático.
- Computação Gráfica: É toda vez que partimos de uma informação, e geramos uma imagem ou uma sequência de imagens. O exemplo claro desta área são as vinhetas realizadas na televisão.
- Processamento de Imagens: É toda vez que partimos de uma "imagem" e criamos uma nova imagem, através de algoritmos que se preocupam somente com a qualidade da imagem. Normalmente não estamos interessados no conteúdo (informação) que ela carrega em si.
- Visão Computacional: É toda vez que partimos de uma "imagem" e nos preocupamos em extrair a informação nela presente, exatamente com é realizado pelo ser humano em seu complexo Sistema Visual. O Objetivo principal desta área é construir a máquina dedicada ao "reconhecimento".

3.4.3 Histograma

Conforme Filho e Neto (1999) os histogramas são utilizados para obter indicações da qualidade de imagem quanto ao nível de contraste, ou conforme seu nível médio caso a imagem seja clara ou escura. O histograma de uma imagem nada mais é do que um conjunto de números que indicam o percentual de pixels na imagem que apresentam um determinado nível de cinza, veja o exemplo da figura 9. Normalmente os histogramas são demonstrados em gráficos de barras, ou em texto com o número ou percentual de pixels correspondentes na imagem.

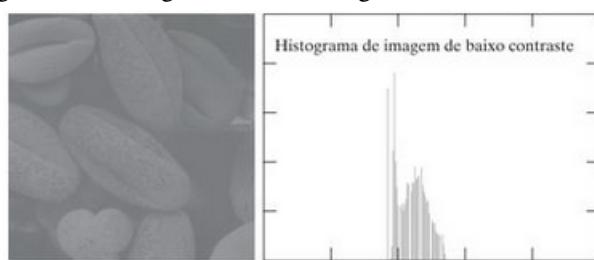
Figura 9 – Histograma em forma de barras.



Fonte – MATHWORKS, 2019a.

Porém caso exista a necessidade de destacar detalhes na imagem a equalização pode ser uma grande aliada representada na figura 10, isso normalmente é feito em imagens para identificação de objetos, imagens de estudos de áreas por satélite e para identificação de padrões em imagens médicas (FILHO; NETO, 1999).

Figura 10 – Histograma de uma imagem com baixo contraste.



Fonte – GOMES, 2019.

3.5 Rede Neural Artificial

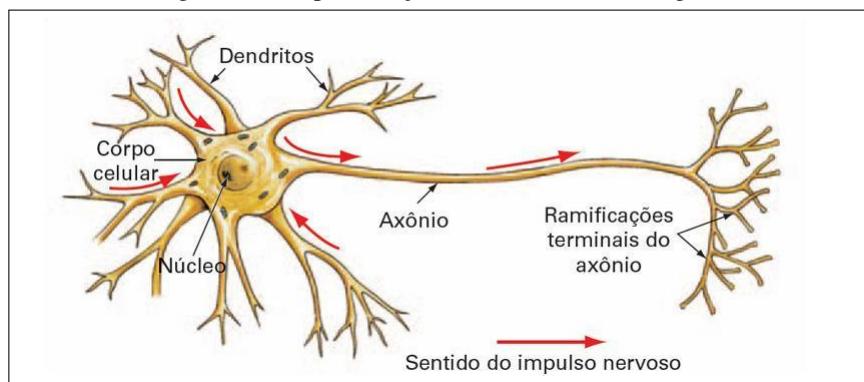
3.5.1 Introdução

Seguinte capítulo introduz os conceitos básicos de uma Rede Neural Artificial (RNA). Explicando a sua origem e os fundamentos de sua estrutura.

3.5.2 Neurônio Biológico

O neurônio é uma estrutura cerebral, especializada na transmissão de informações. A comunicação (sinapse) entre neurônios é realizada através dos dendritos, o corpo celular é o receptor das mensagens captadas por nossos sentidos e conduzida pelo axônio até as suas ramificações, enviando assim, uma suposta resposta ao estímulo captada pelo meio. As informações que são transmitidas pelos neurônios podem ser traduzidas como impulsos elétricos. Tendo como base o neurônio biológico, foi elaborado um modelo matemático baseado no seu princípio de funcionamento que deu origem a construção de redes neurais artificiais. A figura 11 demonstra o neurônio biológico (BRAGA; LUDERMIR; CARVALHO, 2000).

Figura 11 – Representação de um neurônio biológico.

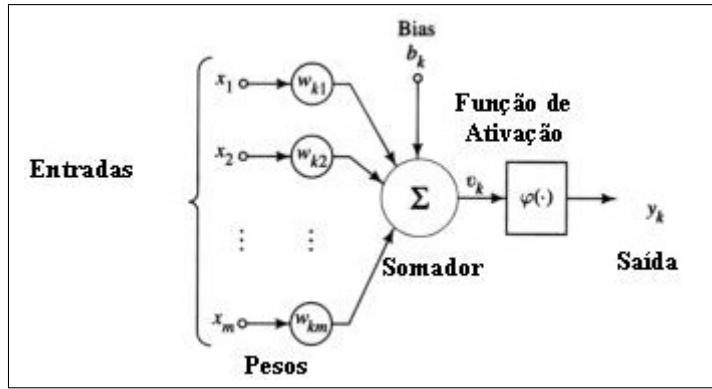


Fonte – BOOK, 2018.

3.5.3 Neurônio Matemático

A partir do conceito de neurônio biológico foi desenvolvido o seu conceito computacional. O modelo Perceptron é uma representação simplificada de um neurônio, permitindo que se tenha uma compreensão de seu funcionamento. O seguinte modelo foi criado por Frank Rosenblatt entre 1950 e 1960, baseando-se nos conceitos determinados por Warren McCulloch e Walter Pitts. A figura 12 demonstra o modelo Perceptron (HAYKIN, 2003).

Figura 12 – Representação de um neurônio matemático.



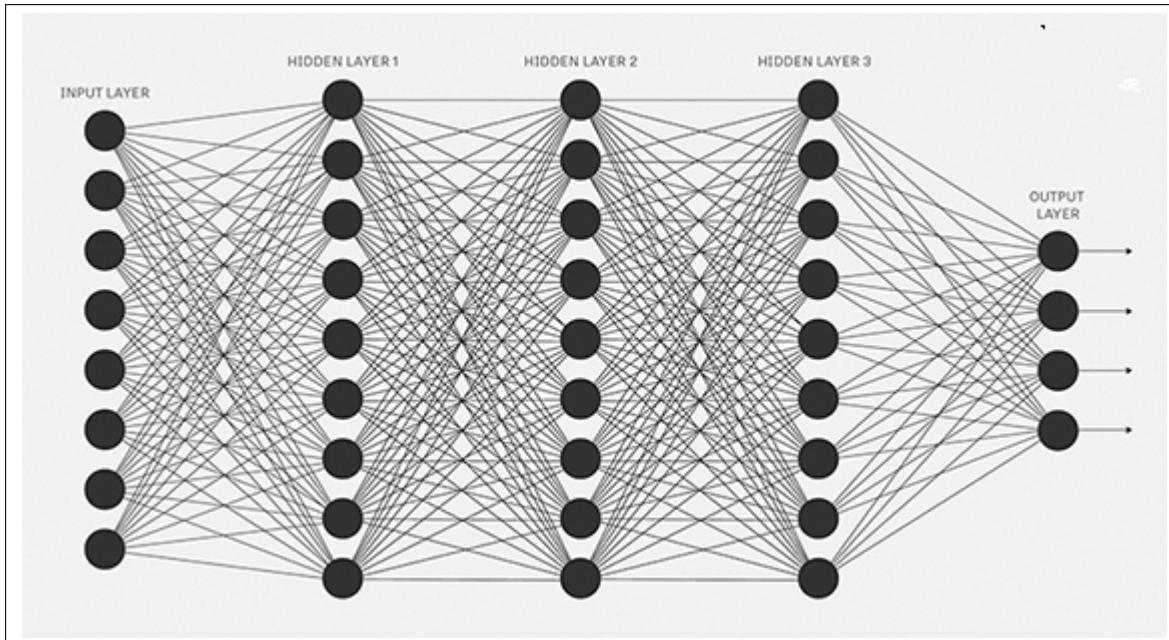
Fonte – UFSC, 2011.

Segundo Braga, Ludermir e Carvalho (2000) essa arquitetura normalmente é utilizada para resolver problemas lineares. O mesmo se baseia no modelo *feedforward*, pois o fluxo que é adotado vai da camada de entrada até a camada de saída. O Perceptron é composto de várias partes sendo elas:

- Sinais de entrada: Recebe o valor a ser treinado.
- Pesos sinápticos: São valores definidos aleatoriamente e responsável por ponderar os valores de entrada.
- Somador: Realiza a soma ponderada das entradas com os pesos.
- Bias: Também chamado de potencial de ativação, tem como finalidade potencializar a ativação de um neurônio.
- Função de Ativação: Seu objetivo é limitar a saída de um neurônio em um intervalo de valores.
- Sinal de Saída: É o resultado determinado pela rede com base nas entradas e pesos.

Esse modelo foi amplamente utilizado pela sua simplicidade, entretanto sua limitação levou ao desenvolvimento de uma arquitetura mais robusta. Com o passar do tempo os problemas foram evoluindo, não sendo mais capaz de resolver com o Perceptron, sentiu-se a necessidade de desenvolver uma arquitetura que não apenas fosse capaz de resolver problemas lineares mas também problemas não-lineares, e assim foi criado o MLP. O MLP é composto de um conjuntos de várias camadas de *Perceptron*, alcançando um alto grau de complexidade, uma vez que a entrada de um neurônio pode influenciar na saída do outro. A figura 13 demonstra a representação de um rede MLP (BRAGA; LUDERMIR; CARVALHO, 2000).

Figura 13 – Representação do modelo MLP.



Fonte – BOOK, 2017.

3.5.4 Funções de Ativação

No processo de treinamento pode ocorrer situações, em que a classificação pode não sair como o esperado, tendo que alterar o valor dos pesos individualmente para que a classificação ocorre. No entanto um problema ocorre, pois alterar o valor dos pesos pode ocasionar na desconfiguração da rede. Entretanto para resolver o impasse sem comprometer a rede utilizamos o conceito de função de ativação. As funções de ativação são transformações não lineares realizadas nas entradas dos neurônios fazendo com que tarefas mais complexas sejam fáceis tanto de executar como de aprender (BOOK, 2017 *apud* V, 2017).

3.5.4.1 Função de Etapa Binária

A função de etapa classifica uma única classe, baseada em limiar ativando a mesma somente, se o valor x estiver acima do determinado, do contrário é desativado. No entanto ela não é tão utilizada quanto aos demais, devido a sua classificação (BOOK, 2017 *apud* V, 2017).

$$f(x) = 1, x \geq 0$$

$$f(x) = 0, x < 0$$

3.5.4.2 Função Linear

A função linear é utilizada na resolução de problemas simples. No entanto por ser linear esta função leva algumas desvantagens a principal delas é o fato da derivada da função ser uma constante, fazendo com que o Backpropagation ao ser executado não possua melhorias (BOOK, 2017 *apud* V, 2017).

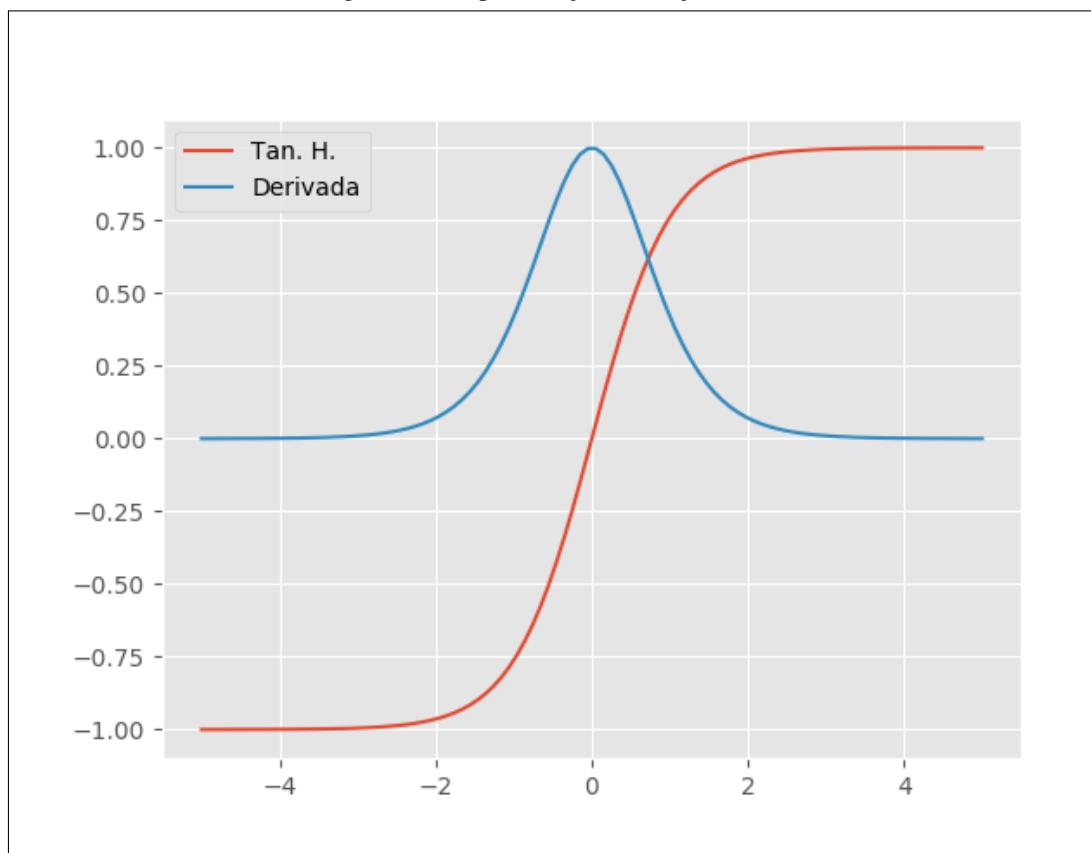
$$f(x) = ax$$

3.5.4.3 Função TanH

A função TanH é uma variação da sigmoide, possuindo as mesmas propriedades, mas com algumas mudanças sendo elas a simetria em relação a origem, tendo assim um intervalo de -1 a 1. A fórmula abaixo é demonstrada na figura 14 a função de ativação TanH() (BOOK, 2017 *apud* V, 2017).

$$\text{TanH}(x) = 2/((1 + e^{(-2x)}) - 1)$$

Figura 14 – Representação da função TanH.



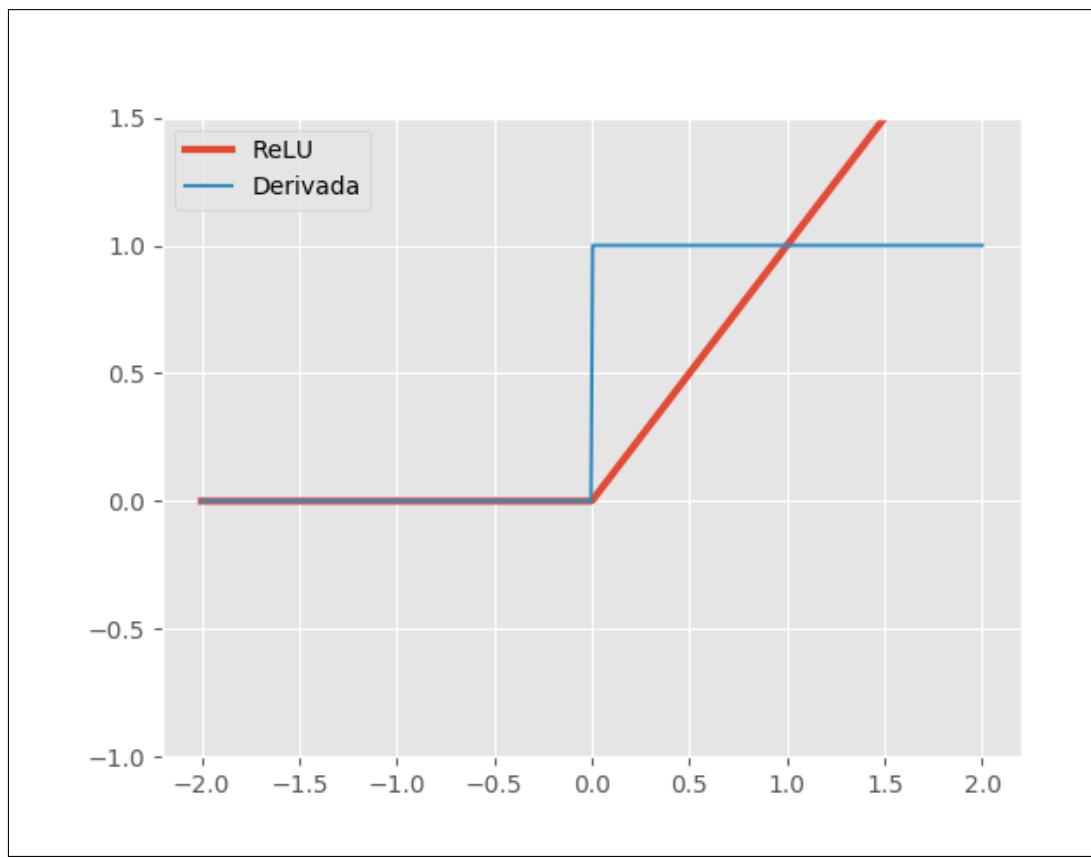
Fonte – FACURE, 2017.

3.5.4.4 Função ReLU

A função ReLU atualmente é a mais utilizada devido a não ser linear, e por não ativar todos os neurônios ao mesmo tempo. A derivada da função é 1 o que deixa rápido, contudo caso tenha pesos muito negativos, fará com que as unidades não sofram atualizações. A fórmula na figura 15 demonstra o gráfico de uma função ReLU() (BOOK, 2017 *apud* V, 2017).

$$f(x) = \max(0, x)$$

Figura 15 – Representação da função ReLU.



Fonte – FACURE, 2017.

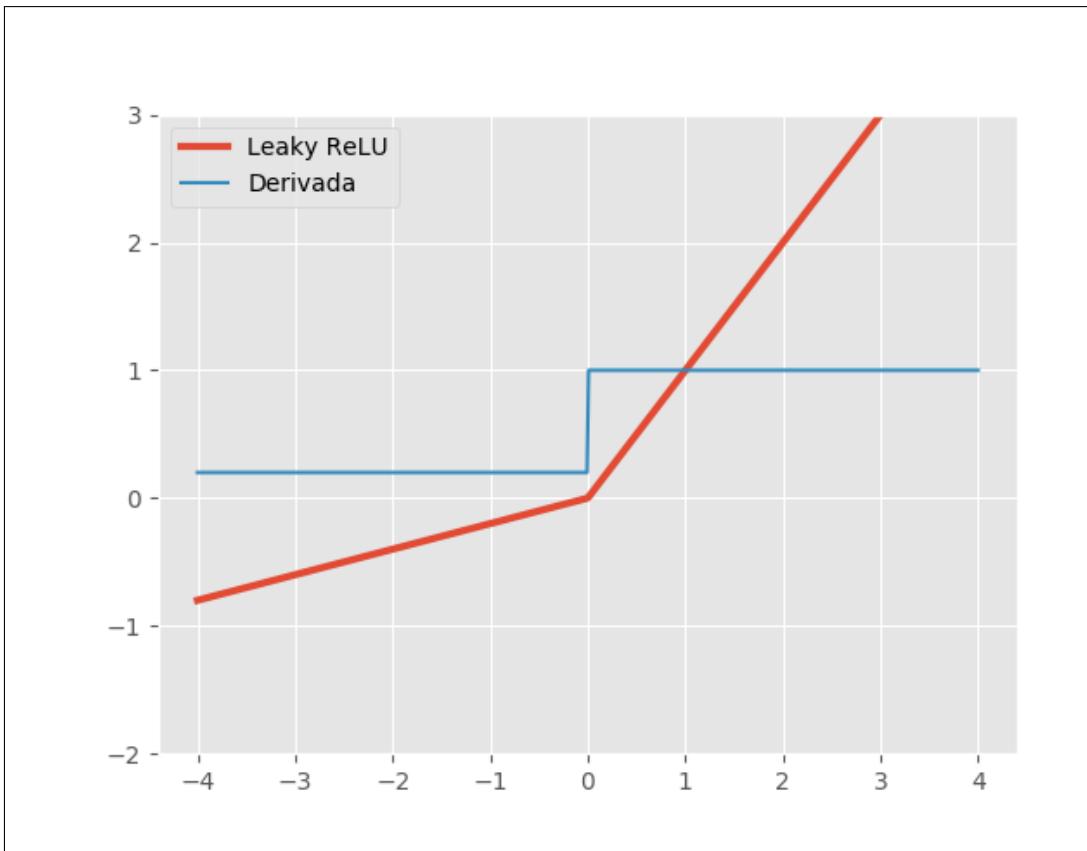
3.5.4.5 Função Leaky ReLU

A função Leaky ReLU consiste em uma variação da ReLU, porém agrupa valores negativos resolvendo assim os problemas dos valores menores que zero. A fórmula na figura 16 demonstra o gráfico de uma função leaky ReLU (BOOK, 2017 *apud* V, 2017).

$$f(x) = ax, x \geq 0$$

$$f(x) = x, x < 0$$

Figura 16 – Representação da função Leaky ReLU.



Fonte – FACURE, 2017.

3.5.4.6 Softmax

O softmax é uma variante como TanH da função sigmoidal, contudo se destaca por lidar com várias classes gerando uma probabilidade a cada uma, modificando a saída de cada classe para valores entre o intervalo de 0 a 1 (BOOK, 2017 *apud* NADA,).

$$\sigma(Z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \text{ for } j = 1, \dots, K.$$

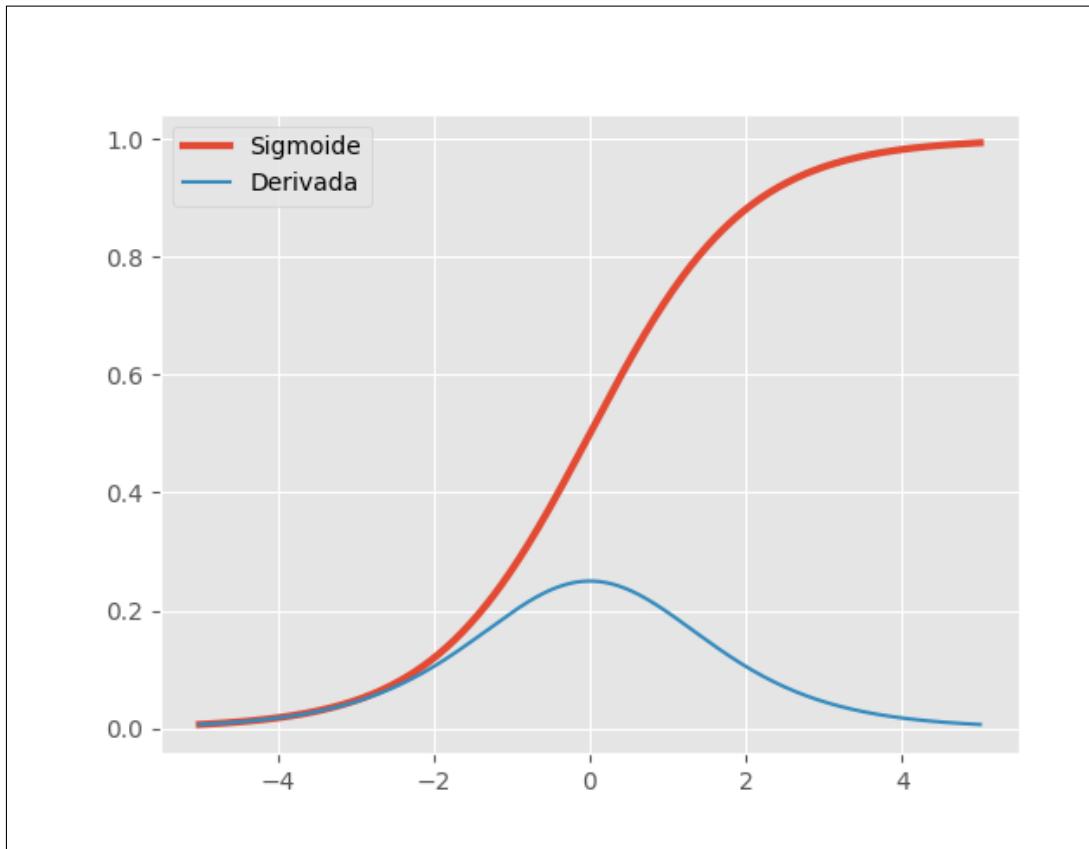
3.5.4.7 Função Sigmoidal

A função sigmoide ou também conhecida como regressão logística, é uma das mais utilizadas, tendo como maior vantagem a não linearidade. Com um intervalo entre 0 e 1, a função essencialmente tenta empurrar os valores de y para os extremos. Mas quando o valor do gradiente é muito baixo tendendo a zero, a aprendizagem não está sendo tão eficiente. Outro problema é seu intervalo não simétrico em relação a origem, fazendo com que os valores não

tenham tanta variedade. A figura 17 demonstra a representação da função sigmoidal (BOOK, 2017 *apud* V, 2017).

$$f(x) = 1 / (1 + e^{-x})$$

Figura 17 – Representação da função Sigmoidal.



Fonte – FACURE, 2017.

3.5.5 Redes Neurais Convolucionais

As Redes neurais convolucionais é um tipo de arquitetura que reconhece quaisquer tipos de objetos em uma imagem e as classifica. Este tipo de arquitetura está presente em conceitos como visão computacional. A rede convolucional é dividido em três objetivos: Extração de características, Mapeamento de características e subamostragem (HAYKIN, 2003).

3.5.5.1 Extração de características

Trabalha mais com as características que define o objeto, desconsiderando assim sua posição na imagem. Desta maneira a análise será feita de forma local (HAYKIN, 2003).

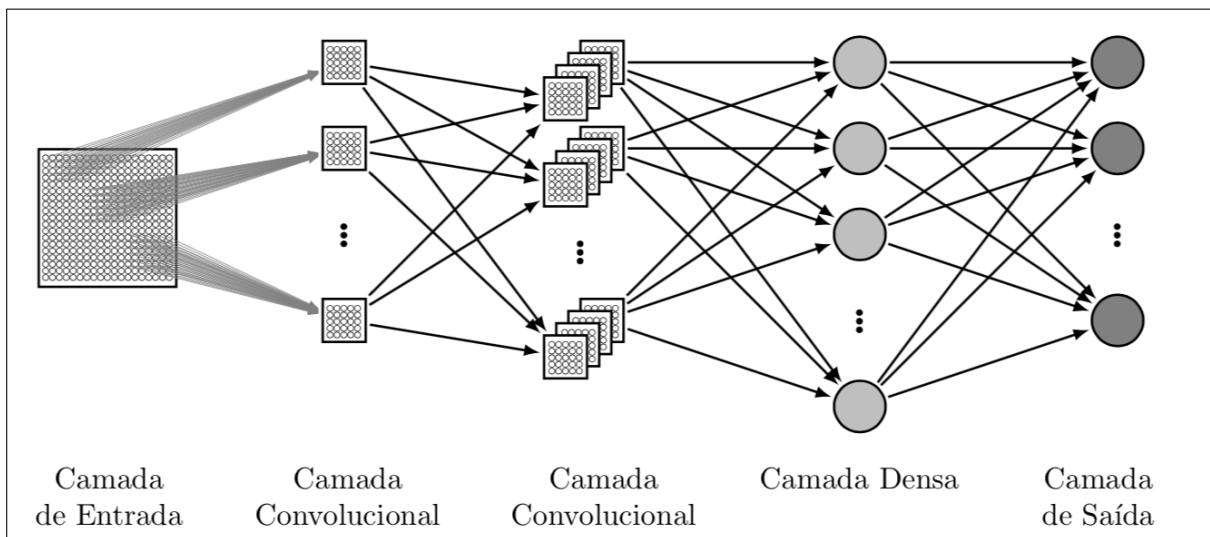
3.5.5.2 Mapeamento de características

Cada camada que compõe a rede é responsável por processar um mapa contendo a característica do seguinte objeto. Os seguintes mapas são denominados de *feature maps* (HAYKIN, 2003).

3.5.5.3 Subamostragem

Após o processo de cada camada de convolução é realizado a etapa de subamostragem ou subsampling capturando as amostras realizadas em cada mapa. O processo de convolução basicamente consiste em multiplicação entre as matrizes de filtro e mapa representado na figura 18 (HAYKIN, 2003).

Figura 18 – Representação de uma rede convolucional.



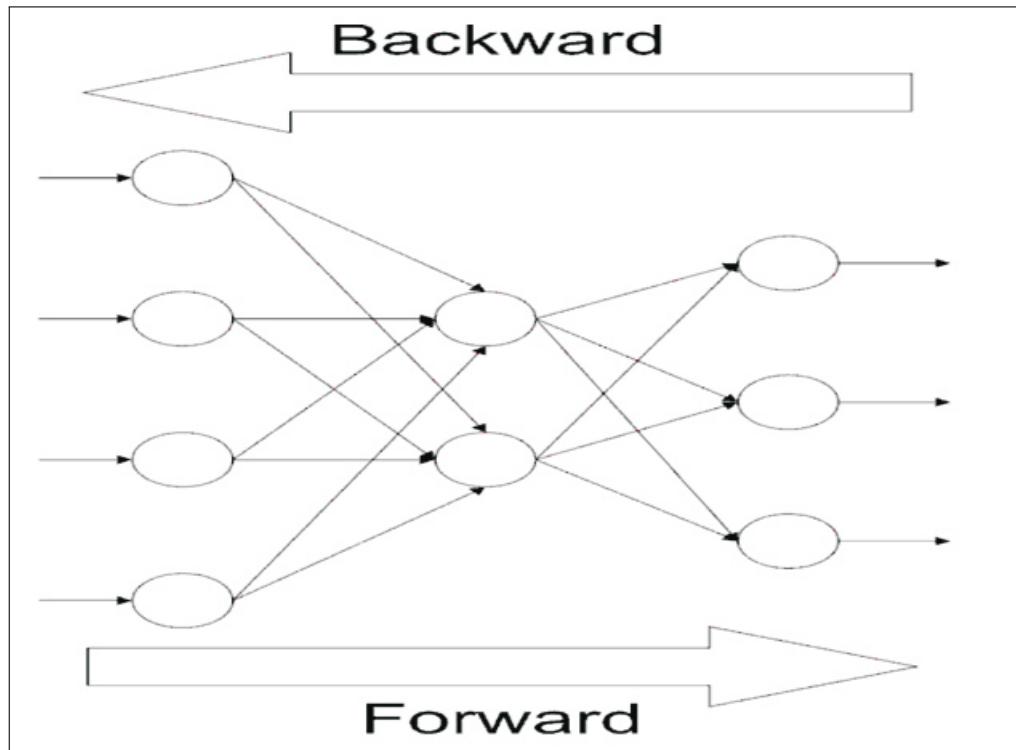
Fonte – SAKURAI, 2017.

3.5.6 Backpropagation

Em uma rede MLP é comumente utilizado o algoritmo backpropagation para auxiliar no treinamento, como foi mencionado no tópico anterior. O sentido de como os dados percorre uma rede MLP é da camada de entrada até a camada de saída. Esse fluxo é denominado de forward, esta primeira fase visa ter resultado da rede com base nas entradas estipuladas. Em seguida é feita uma comparação dos dados gerados pela rede MLP com os dados esperados, se caso o resultado da comparação seja equivalente significa então que o treinamento da rede foi um sucesso, caso contrário é feita uma segunda fase nomeado de *backward*, onde ocorre o *backpropagation*. O *Backpropagation* é um algoritmo, cuja a função é reajustar os pesos da

rede, para realizar esse reajuste o algoritmo se baseia na diferença entre os valores esperados e os valores obtidos pelo treinamento. O resultado desse cálculo é denominado de erro, e com esse erro que feito o reajuste. Após o reajuste é feito novamente o ciclo de *forward* e *backward* até que o resultado seja alcançado. A figura 19 demonstra o fluxo do *backpropagation* (SILVA; SPATTI, 2016).

Figura 19 – Fluxo de aprendizado do algoritmo *propagation*.



Fonte – NAKAMURA, 2013.

3.6 Ferramentas

3.6.1 Introdução

O capítulo apresenta o conceito das ferramentas utilizadas.

3.6.2 OpenCV

"O Open Source Computer Vision Library (OpenCV) é uma biblioteca em código aberto de visão computacional e aprendizado de máquina. OpenCV foi construído para fornecer infraestrutura comum para aplicativos de visão computacional e para acelerar o uso da percepção da máquina nos produtos comerciais." (OPENCV, 2019).

Possui integração com ambientes preparados para trabalhar com diversas linguagens como C++, Python, Java e MATrix LABoratory (MATLAB) em ambientes Windows, Linux,

Android e Mac OS (OPENCV, 2019).

3.6.3 Python

A surpreendente linguagem script foi criada pelo Holandês Guido van Rossum. Ela é um conjunto unificado de experiências que Guido vivenciou em projetos anteriores como a linguagem ABC assim chamada no Centrum voor Wiskunde en Informatica (CWI) (VENNERS, 2003).

“Guido van Rossum: Em 1986, mudei para um projeto diferente no CWI, o projeto Amoeba. Amoeba era um sistema operacional distribuído. No final dos anos 80, descobrimos que precisávamos de uma linguagem script. Eu tive um grande grau de liberdade nesse projeto para iniciar meu próprio mini projeto dentro do escopo do que estávamos fazendo. Lembrei-me de toda a minha experiência e u poço da frustração com a ABC. Eu decidi tentar criar uma linguagem de script simples que possuísse algumas das melhores propriedades do ABC, mas sem seus problemas.” (VENNERS, 2003).

Trata-se de uma linguagem open source com uma grande comunidade respeitada que contribui e mantém sempre atualizada documentação ou módulos novos (2, 2003).

A linguagem tem um grande histórico evolutivo tanto em seus módulos quanto em popularidade. Atualmente ocupa o 3º lugar no ranking de popularidade, portanto este registro de colocação é o maior apresentando desde 2001 (BV, 2019).

3.6.4 MATLAB

O MATLAB é um *software* que usa diferentes algoritmos para tratar dados de entrada para obter a saída com os resultados desejados usando análise interativa com uma linguagem de programação que gera uma matriz numérica dos dados inseridos assim podendo ser trabalhada no algoritmo a ser implementado (MATHWORKS, 2019b).

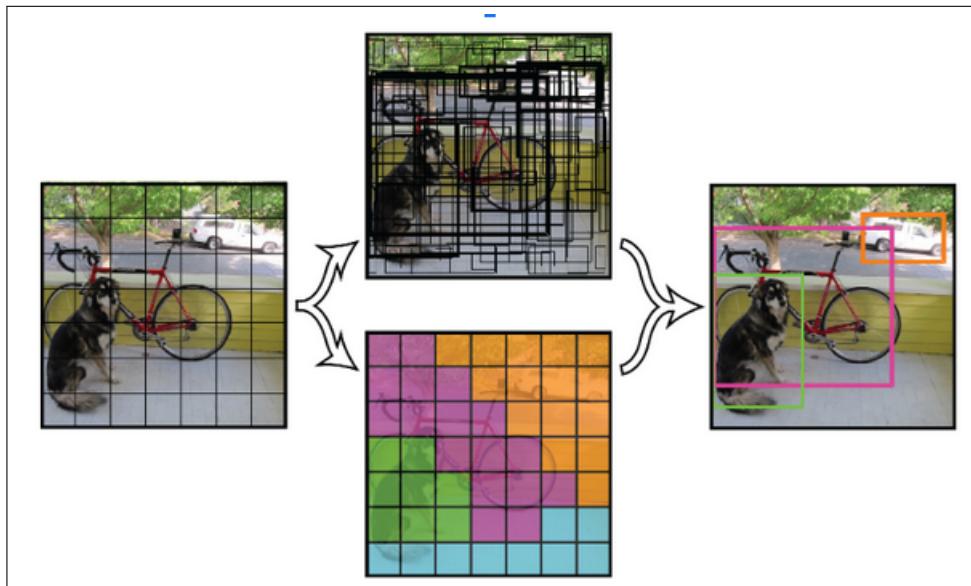
3.6.5 YOLO

O YOLOv3 aplica uma única Rede Neural em toda a imagem , dividindo a imagem em regiões e prevê caixas delimitadoras e probabilidade para cada região, sendo as caixas delimitadoras ponderadas pelas probabilidades previstas. O YOLO é 1000x mais rápido que o Regiões com redes neurais convolucionais (R-CNN) e 100x mais rápido que o *Fast R-CNN* (redes convolucionais rápidas baseadas na região para detecção de objetos), analisando a imagem inteira

com uma única avaliação de rede diferente dos sistemas R-CNN e o *Fast R-CNN* (REDMON *et al.*, 2018).

Segundo Redmon *et al.* (2015) o modelo YOLO básico processa imagens em tempo real a 45 quadros por segundo e sua versão menor da rede, o Fast YOLO, processa surpreendentes 155 quadros por segundo, enquanto ainda alcança o dobro do *mean Average Precision* (*mAP*) de outros detectores em tempo real.

Figura 20 – Processo para reconhecimento de objetos.



Fonte – REDMON, 2019.

A figura 20 demonstra um exemplo de decretação gerada com o YOLOv3. A função Leaky ReLU consiste em uma variação da ReLU, porém agrega valores negativos resolvendo assim os problemas dos valores menores que zero. A fórmula abaixo demonstra o gráfico de uma função leaky ReLU() (BOOK, 2017 *apud* V, 2017).

4 DESENVOLVIMENTO

4.1 Delimitação do Problema

Este projeto tem como intuito de trabalhar de forma engajada a área de serviços de resgate. Seu objetivo é auxiliar o profissional (socorrista) em situações de resgate em ambientes hostis.

Visando auxiliar as buscas este projeto tem como finalidade implementar uma solução para que, pessoas que possuem algum percentual de vida sejam encontradas e recebam o devido atendimento assim resgatando o maior números de vidas possível, utilizando técnicas de processamento de imagem e visão computacional e redes neurais para reconhecer pessoas vivas através de um agente inteligente a ser desenvolvido.

Sendo como requisito imagens térmicas (termogramas) e imagens do espectro visível ao olho humano para classificar o *status* de vida de acordo com padrões treinados.

Com a realização deste projeto será possível resgatar mais vidas direcionando os socorristas nos locais onde as vítimas se encontram.

4.1.0.1 Visão Inicial

Ao realizar o processo de desenvolvimento necessitamos de duas RNA onde uma delas deverá ser capaz de reconhecer pessoas e a outra capaz de identificar se a pessoa está viva através do termograma da imagem.

Para o desenvolvimento criamos um *Workspace* e montamos uma bases de dados com bases de imagens extraídas da internet de diversas fontes, essas imagens serviram para treinar a RNA para reconhecer os padrões que necessitamos.

O algoritmo a ser desenvolvido irá tratar imagens estáticas ao invés da imagem dinâmicas obtida do hardware da câmera em tempo real pois o custo para adquirir o equipamento é extremamente elevado variando de R\$ 2.299,00 a R\$ 63.490,00 dependendo do modelo. Veja as três citações apresentados na figura 21, 22 e na figura 23.

Figura 21 – Modelo de câmera de menor valor.



Camera Termica ON PRO USB-C para Smartphone Android FLIR
2299,00 R\$ [Loja do Mecânico](#)
★★★★★ 218 comentários do produto
 Outras opções de estilos: [FLIR um pro - ios](#) (2250 R\$) [Mais](#)

Fonte – MECÂNICO, 2019.

Figura 22 – Modelo de câmera de valor médio.



Câmera Térmica Sensor Térmico Drone Radiométrico Duo R Flir Duo R Para Drones
R\$ 16.000,00 [Shoptime](#) | [Comparar preços de 2 lojas](#)
 Câmera térmica Sensor Térmico Drone Radiométrico Duo R Flir Duo R para Drones Descrição Breve do Produto:
 Imagens de luz ...

Fonte – SHOPTIME, 2019.

Figura 23 – Modelo de câmera de maior valor.



Câmera Dji Zenmuse Xt2 Térmica Inspeção Para Drone Matrice
63 490,00 R\$ [Mercado Livre](#)
 VISÃO DUPLA. INTELIGÊNCIA SUPERIOR.JUNTE-SE À PRÓXIMA GERAÇÃO DE SOLUÇÕES DE DRONES COMERCIAIS
 COM O XT2.EMPARELHAR O SENSOR ...

Fonte – LIVRE, 2019.

4.2 Engenharia de Software

4.2.1 Introdução

O presente tópico esclarece a Gerência de Escopo exercida sobre o sistema. Fora utilizada tal gerência para de informar os objetivos do projeto aos envolvidos; para que tenha uma comunicação entre os integrantes do projeto. As etapas desta gerência nortearam a regra de negócio do *software* em questão. Estudos sobre viabilização do *software* foram realizados levando em consideração a gerência do tempo, recursos, custos, ferramentas e a descrição dos requisitos elicitados para os resultados esperados.

4.2.2 Gerência de Escopo

Para o qualificar e mensurar cada parte do projeto foram realizados alguns passos para alcançar os objetivos, veja abaixo uma prévia dos seguintes tópicos:

- Elicitação de Requisitos: processo de coleta, validação, classificação e avaliações de requisitos funcionais e não funcionais;
- Projeto Lógico: onde será representado em formas de fluxos (fluxograma) e diagramas (atividade e casos de Uso) cada roteiro de solução de um problema.

Portanto, este tópico aborda todas características, funções, tarefas ou procedimentos principais que constitui o *software*.

4.2.2.1 Requisitos Funcionais

[RF01] O sistema deve manter um histograma como base de referência constituída por imagens agrupadas por pessoas vivas;

[RF02] O sistema deve manter um histograma como base de referência constituída por imagens agrupadas por pessoas que vieram a óbito;

[RF03] O sistema deve manter uma base sólida com percentual de treino adequado a realidade, para obter um bom resultado sobre a análise para que seja correspondente ao que os olhos humanos enxergam;

[RF04] O sistema deve disponibilizar meio para postar imagem como entrada no sistema para efetuar análise sobre o estado da pessoa que se encontra na imagem;

4.2.2.2 Requisitos Não Funcionais

[RNF01] O sistema não deve fazer integração direta com dispositivo câmera;

[RNF02] O sistema não deve realizar avaliação do quadro clínico do indivíduo;

[RNF03] O sistema não deve qualificar indivíduo da foto quanto ao gênero, cor de pele ou até mesmo idade;

[RNF04] O sistema não deve qualificar se o que apresenta na foto é um objeto ou um animal;

[RNF05] O sistema deverá integrar com a biblioteca YOLOv3 para o processo de identificação de um ser humano no ambiente da imagem;

[RNF06] O sistema deverá ser executada em qualquer plataforma desktop.

[RNF07] O sistema deverá ser escrito usando a linguagem de programação Python.

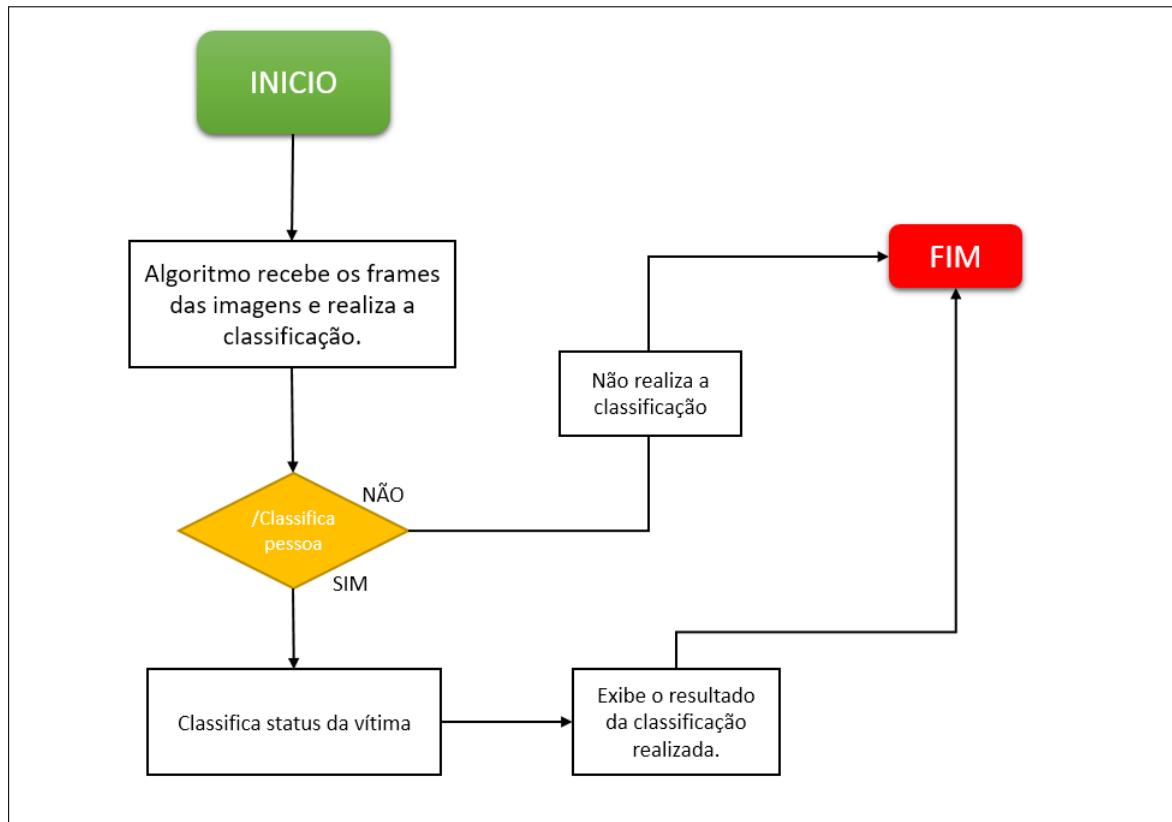
[RNF08] O sistema deve manter a estrutura da análise constituída por RNA e IA;

[RNF09] O sistema deve manter uma análise de ‘julgamento’ com base em estudos sobre espectro, luz e cor que constitui a imagem;

4.2.2.3 Diagrama de Atividades

A figura 24 demonstra o diagrama de atividades do procedimento realizado pelo algoritmo.

Figura 24 – Diagrama de Atividades.

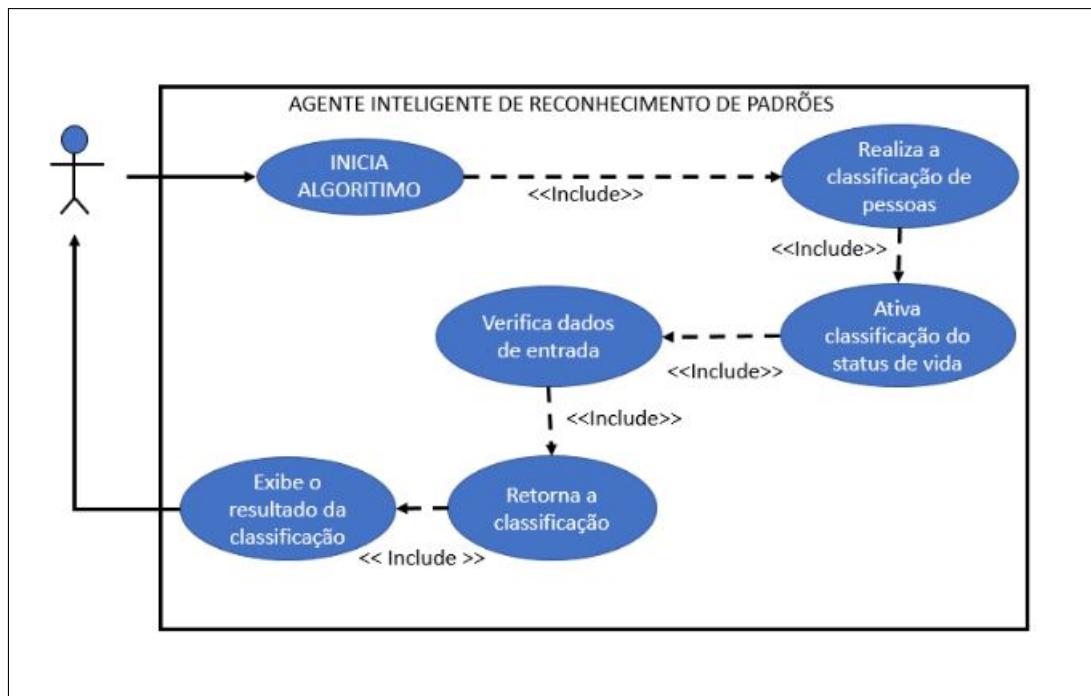


Fonte – Autor, 2019.

4.2.2.4 Diagrama Caso de Uso

A figura 25 exemplifica o caso de uso com interação com o usuário.

Figura 25 – Diagrama Caso de Uso.



Fonte – Autor, 2019.

4.2.3 Gerência do Tempo

O resultado de cada pesquisa realizada possibilitou montar uma margem de tempo para a produção de cada etapa do *software* em questão, desde a documentação até a produção. Veja a figura 26 com o cronograma de atividade como ficou entendido e distribuído por mês conforme os módulos do trabalho.

Figura 26 – Cronograma de Atividade durante o ano de 2019(Tabela).

TÍTULOS	FEVEREIRO	MARÇO	ABRIL	MAIO	JUNHO	JULHO	AGOSTO	SETEMBRO	OUTUBRO	NOVEMBRO	DEZEMBRO
ESCOLHA DO TEMA											
ORIENTADOR											
COLETA DE DADOS											
REFERENCIAL TEÓRICO											
DESENVOLVIMENTO											
REDAÇÃO											
REVISÃO ABNT											

Fonte – Autor, 2019.

Este cronograma veio para o estabelecer metas de entregas. No intuito de garantir a entrega do trabalho dentro do prazo estabelecido.

4.3 Configuração de Ambientes

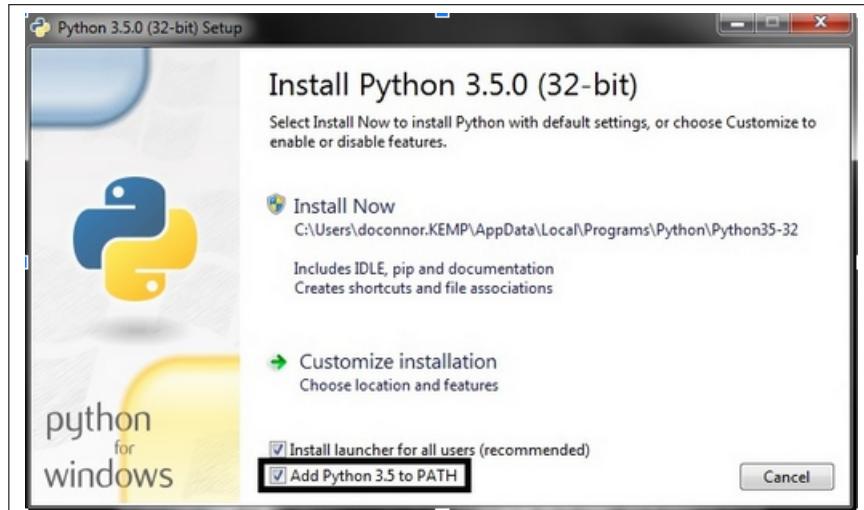
4.3.1 Introdução

Neste capítulo seguirá passo a passo da instalação das ferramentas necessárias para o desenvolvimento . Para preparar a montagem do ambiente de trabalho foi necessário instalar os pacotes e dependências para o funcionamento do algoritmo.

4.3.2 Instalando Python

Como a linguagem de programação usada foi python foi instalado a versão Python 3.5.0 com todos os pacotes versão obtida no site:([https://www.python.org/downloads/release/python-350.](https://www.python.org/downloads/release/python-350/)).

Figura 27 – Instalação do Python 3.5.0 (32-bit).



Fonte – 1, 2019.

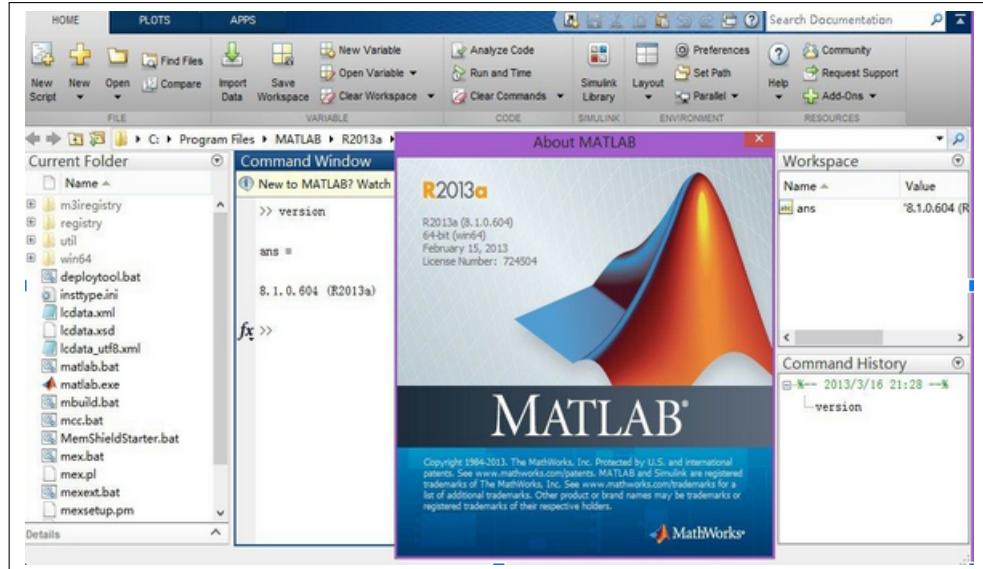
Nessa parte da instalação deve se marcar a opção “Add Python 3.5 to Path” e após seguir com a instalação normal como demonstra a figura 27.

4.3.3 Instalando MATLAB

Para o desenvolvimento da rede foi utilizado o *software* do MatLab disponibilizado pela faculdade, mas o mesmo se encontra disponível no link:

<https://www.mathworks.com/products/matlab.html>.

Figura 28 – Instalação do MATLAB.



Fonte – Autor, 2019.

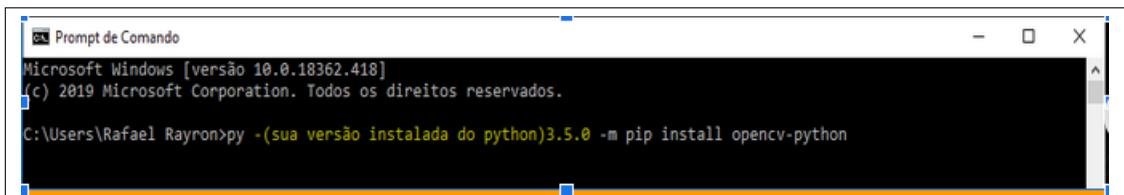
Veja na figura 28 acima como é a interface da tela inicial do programa Matlab.

4.3.4 Bibliotecas

4.3.4.1 Instalando OpenCV

Depois do Python instalado seguimos com a instalação do openCV da seguinte forma:
Executamos o Prompt de comandos e passamos o seguinte comando exemplificado na figura 29.

Figura 29 – Instalação da biblioteca OpenCV.



Fonte – Autor, 2019.

A figura 29 demonstra como realizar a instalação e exemplifica a sintaxe do comando. No campo em amarelo na imagem deve-se colocar o número da versão do Python instalada no tópico anterior. Após concluir a instalação será exibida uma mensagem de sucesso.

4.3.4.2 Instalando Numpy

De todas as bibliotecas usadas a única necessária instalação foi a “numpy”, pois as demais foram instaladas nos passos anteriores. A biblioteca numpy foi instalada com o seguinte comando presente na figura 30.

Figura 30 – Instalação da biblioteca Numpy.

Fonte – Autor, 2019.

A biblioteca basicamente é usada para realizar cálculos em Arrays Multidimensionais em Arrays.

4.4 Roteiros de Treinamentos RNA

4.4.1 Introdução

O capítulo abordará a bateria de treinamentos realizados no percurso do presente trabalho. Será explorado detalhes como desafios, erros e acertos em cada etapa. Quanto a usabilidade das funções, também será apresentado os módulos do MATLAB utilizados no processo de busca pela RNA recomendada ao projeto.

4.4.2 Preparando a base de dados para treino

Para realizar a extração de dados da imagem foi desenvolvido um algoritmo em python onde o mesmo extrai o histograma da imagem e salva em um arquivo formatado em “.csv” para ser utilizado no MatLab. O código representado na imagem abaixo foi utilizado a função “*matplotlib.pyplot.hist*” que faz a extração do histograma da imagem, essa função pertence a biblioteca *matplotlib* e serve para calcular e desenhar o histograma e sendo o seu valor de retorno uma tupla.

Figura 31 – Código para extrair o histograma da imagem em python.

```

1 import csv
2 import cv2
3 import numpy as np
4 from matplotlib import pyplot as plt
5
6 ▼ with open("inputs.csv",'w',newline='') as saida:
7     escrever = csv.writer(saida)
8     ▼ for i in range(1,40):
9         print(str(i)+".jpg")
10        img = cv2.imread(str(i)+".jpg")
11        matriz = plt.hist(img.ravel(),256,[0,256])
12        escrever.writerow(matriz[0])
13        print(matriz[0])
14

```

Fonte – Autor, 2019.

Após a extração do histograma das imagens obteve-se o resultado abaixo onde cada linha são características de um termograma do total de 40 termogramas de pessoas vivas geradas.

Figura 32 – Representação das imagens, cada linha é uma imagem.

	A	B	C	D	E	F	G	H	I	J	K	L
1	21242.0	8363.0	3012.0	1557.0	1152.0	876.0	832.0	654.0	677.0	619.0	588.0	565.0
2	23745.0	9625.0	2944.0	1238.0	866.0	645.0	563.0	543.0	458.0	429.0	426.0	394.0
3	40023.0	17200.0	4730.0	2077.0	1388.0	1022.0	898.0	805.0	756.0	736.0	668.0	654.0
4	22998.0	9863.0	3547.0	1612.0	1088.0	803.0	596.0	561.0	491.0	567.0	683.0	701.0
5	15281.0	5873.0	2007.0	854.0	544.0	436.0	418.0	358.0	319.0	312.0	300.0	294.0
6	17045.0	7807.0	2997.0	1403.0	856.0	606.0	502.0	423.0	494.0	467.0	379.0	377.0
7	29303.0	13208.0	4711.0	2011.0	1342.0	944.0	825.0	725.0	690.0	631.0	644.0	644.0
8	166621.0	6468.0	2063.0	899.0	598.0	442.0	440.0	358.0	335.0	301.0	277.0	229.0
9	27680.0	12228.0	3786.0	1737.0	1255.0	931.0	788.0	719.0	609.0	577.0	554.0	548.0
10	25286.0	10537.0	2551.0	876.0	458.0	417.0	326.0	313.0	287.0	306.0	296.0	360.0
11	30937.0	12960.0	3986.0	1632.0	1051.0	799.0	646.0	602.0	788.0	791.0	1311.0	1051.0
12	9588.0	3876.0	636.0	143.0	69.0	47.0	59.0	35.0	35.0	36.0	27.0	24.0
13	7894.0	3302.0	511.0	121.0	101.0	89.0	82.0	77.0	64.0	76.0	58.0	82.0
14	12393.0	5554.0	1454.0	386.0	181.0	127.0	105.0	106.0	80.0	75.0	65.0	63.0
15	26962.0	11142.0	2943.0	1203.0	857.0	703.0	635.0	603.0	711.0	632.0	632.0	638.0
16	36642.0	18440.0	3828.0	1566.0	1081.0	906.0	769.0	705.0	651.0	653.0	649.0	551.0
17	35894.0	13391.0	4941.0	2557.0	1603.0	1308.0	1102.0	1011.0	862.0	842.0	808.0	852.0
18	46910.0	19106.0	2650.0	1279.0	924.0	695.0	617.0	552.0	516.0	497.0	472.0	416.0

Fonte – Autor, 2019.

Como as entradas no MatLab para o treinamento é por colunas então foi realizado a transposta da matriz acima onde cada coluna passa a representar as características de uma uma imagem.

Figura 33 – Representação das imagens de forma transposta, cada coluna é uma imagem.

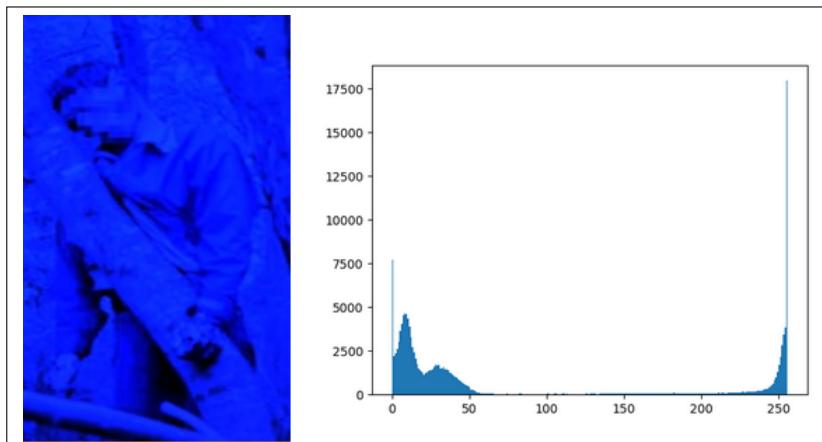
	A	B	C	D	E	F	G	H	I	J	K
1	21242.0	23745.0	40023.0	22998.0	15281.0	17045.0	29303.0	16621.0	27680.0	25286.0	30937.0
2	8363.0	9625.0	17200.0	9863.0	5873.0	7807.0	13208.0	6468.0	12228.0	10537.0	12960.0
3	3012.0	2944.0	4730.0	3547.0	2007.0	2997.0	4711.0	2063.0	3786.0	2551.0	3986.0
4	1557.0	1238.0	2077.0	1612.0	854.0	1403.0	2011.0	899.0	1737.0	876.0	1632.0
5	1152.0	866.0	1388.0	1088.0	544.0	856.0	1342.0	598.0	1255.0	458.0	1051.0
6	876.0	645.0	1022.0	803.0	436.0	606.0	944.0	442.0	931.0	417.0	799.0
7	832.0	563.0	898.0	596.0	418.0	502.0	825.0	440.0	788.0	326.0	646.0
8	654.0	543.0	805.0	561.0	358.0	423.0	725.0	358.0	719.0	313.0	602.0
9	677.0	458.0	756.0	491.0	319.0	494.0	690.0	335.0	609.0	287.0	788.0
10	619.0	429.0	736.0	567.0	312.0	467.0	631.0	301.0	577.0	306.0	791.0
11	588.0	426.0	668.0	683.0	300.0	379.0	644.0	277.0	554.0	296.0	1311.0
12	565.0	394.0	654.0	701.0	294.0	377.0	644.0	229.0	548.0	360.0	1051.0
13	542.0	411.0	769.0	468.0	272.0	429.0	609.0	221.0	465.0	320.0	671.0
14	520.0	447.0	650.0	388.0	240.0	382.0	538.0	254.0	479.0	318.0	570.0
15	531.0	377.0	634.0	387.0	207.0	393.0	549.0	245.0	419.0	300.0	438.0
16	535.0	337.0	621.0	369.0	226.0	347.0	550.0	225.0	405.0	236.0	386.0
17	466.0	379.0	662.0	357.0	215.0	320.0	505.0	218.0	391.0	270.0	372.0

Fonte – Autor, 2019.

O mesmo processo realizado para o termograma de pessoas mortas um total de 12 termogramas.

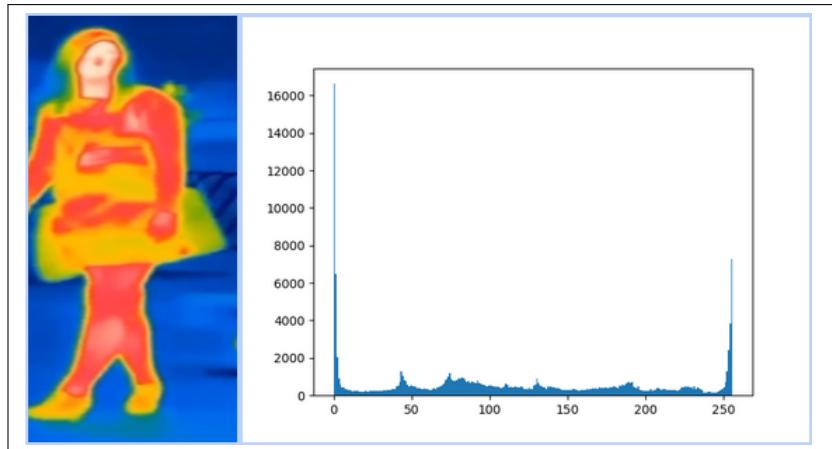
Após a realização do processo foi extraído 256 características de cada termograma para ser imputadas no Matlab, exemplificando nas figuras 34 e 35.

Figura 34 – Histograma de imagem com pessoa morta.



Fonte – Autor, 2019.

Figura 35 – Histograma de imagem com pessoa viva.



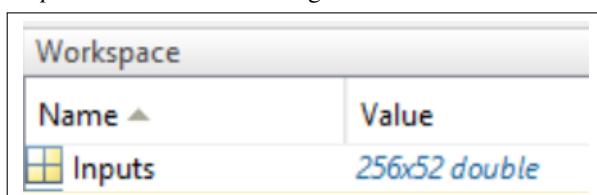
Fonte – Autor, 2019.

Pela dificuldade de não encontrar termogramas que exemplifique o uso em ambientes hostis foi utilizado e gerado como base de dados imagens aproximadas a uma câmera real.

4.4.3 Criando RNA no MATLAB

Para criar e treinar a rede neural utilizou-se a ferramenta *Neural Pattern Recognition* no Matlab disponível na aba aplicações do *software*, mas o primeiro passo a ser realizado é separar a base de características da imagem e então geramos um arquivo a ser imputado para treino no *workspace* do Matlab. A figura 36 faz parte do workspace onde 52 *Inputs* representam a quantidade de imagens que foi dado como entrada, já os 256 em *Inputs* representam as características de cada uma.

Figura 36 – *Workspace* - Entrada de 52 imagens com 256 características em cada uma.



Fonte – Autor, 2019.

O próximo passo é gerar um arquivo de saída pois como o treino é supervisionado é preciso passar as entradas e quais são as saídas. A figura 37 traz a representação de quantidade de saídas para as 52 imagens que foram inseridas neste *workspace*.

Figura 37 – *Workspace* - Arquivo de saída para as 52 imagens gerando 2 saídas possíveis.

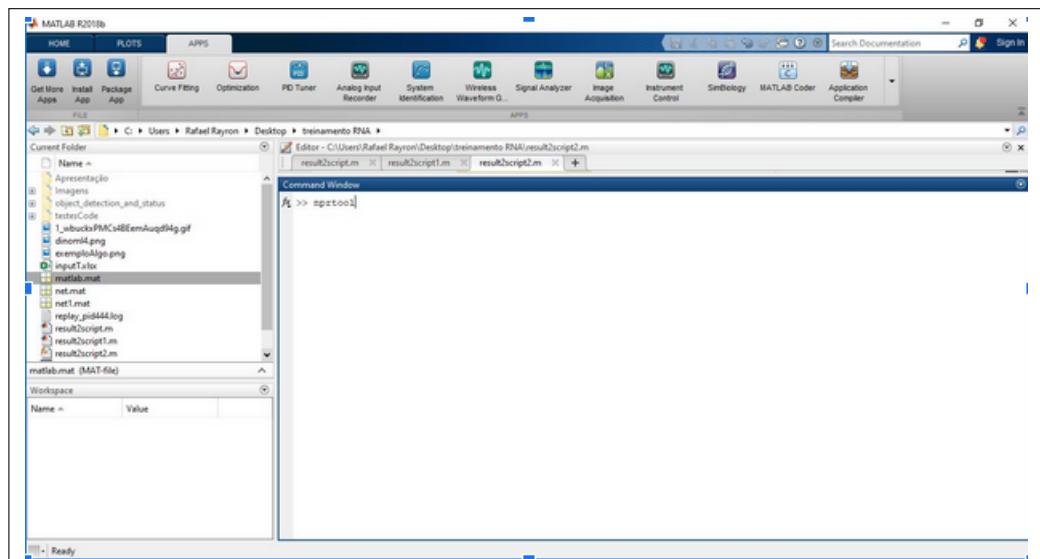
Name	Value
Inputs	256x52 double
Targets	2x52 double

Fonte – Autor, 2019.

Basicamente rede a ser criada terá a capacidade de classificar as entradas em um conjunto de categorias de destino. Utilizando o aplicativo *Neural Pattern Recognition* será selecionado os dados mencionados acima, para criar e treinar uma rede e avaliar seu desempenho usando matrizes de entropia cruzada e confusão.

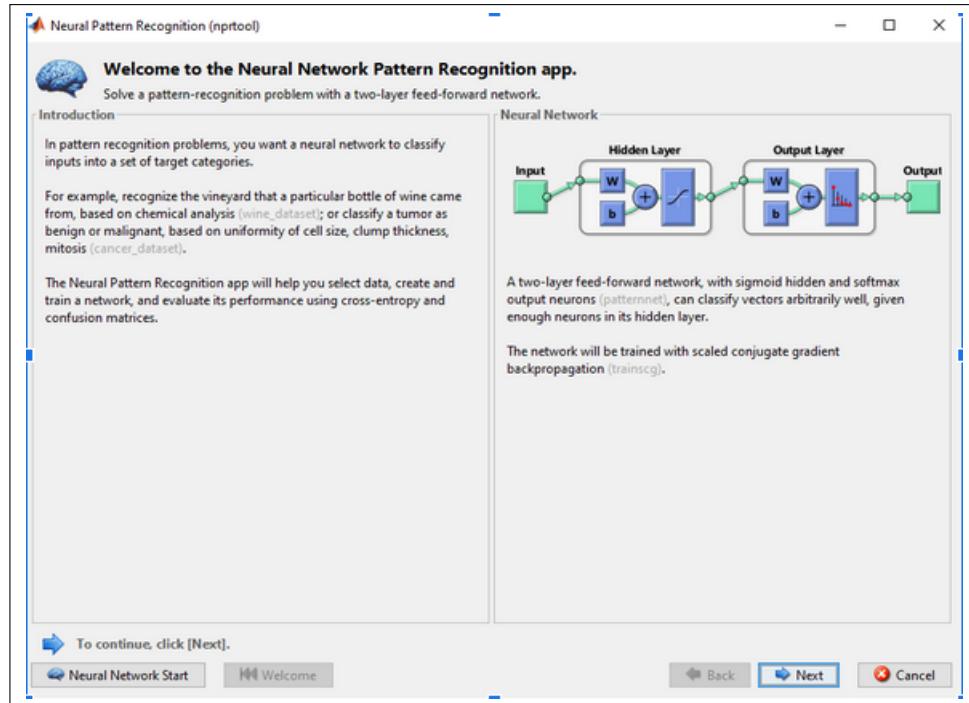
Passando o comando *NFTool* para abrir a api de treinamento como mostra a figura 38. Ao enviar o comando é aberta uma tela explicativa do *Neural Pattern Recognition* como mostra a figura 38, basta clicar em *next* para continuar o procedimento.

Figura 38 – *NFTool* - Inserindo comando.



Fonte – Autor, 2019.

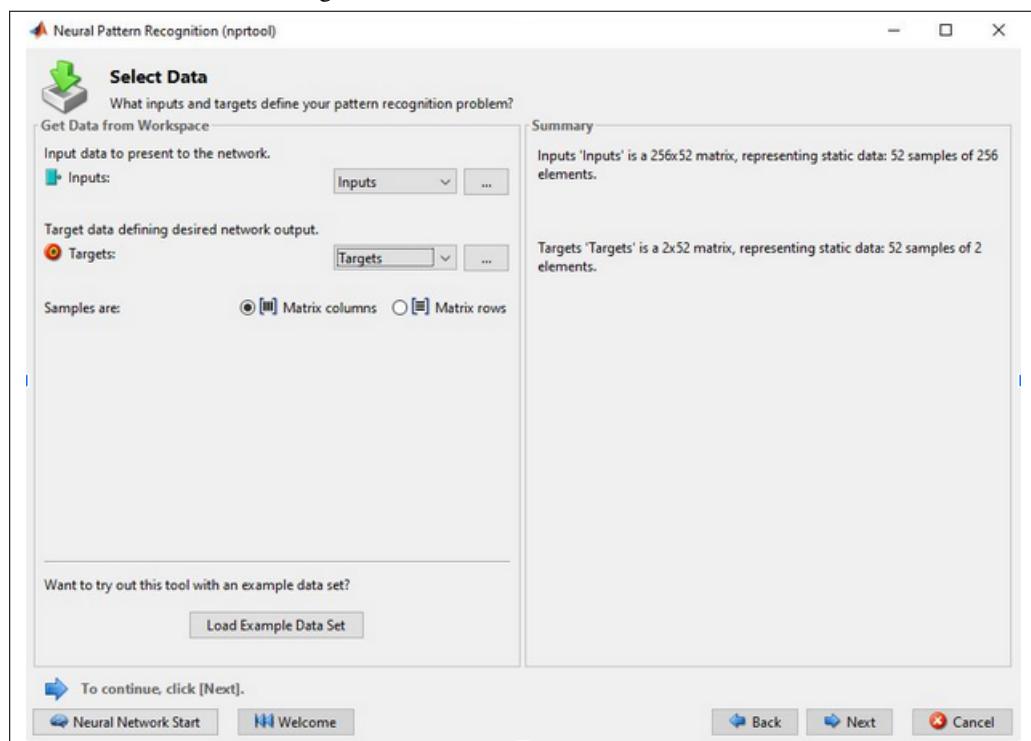
Figura 39 – NFTool - Introdução do processo.



Fonte – Autor, 2019.

Na etapa seguinte, é necessário passar as imagens que selecionamos para treinar a rede neural. Sendo que o *input* representando os termogramas e o *output* as saídas geradas sendo com *status* de sem vida ou com vida demonstrado na figura 40.

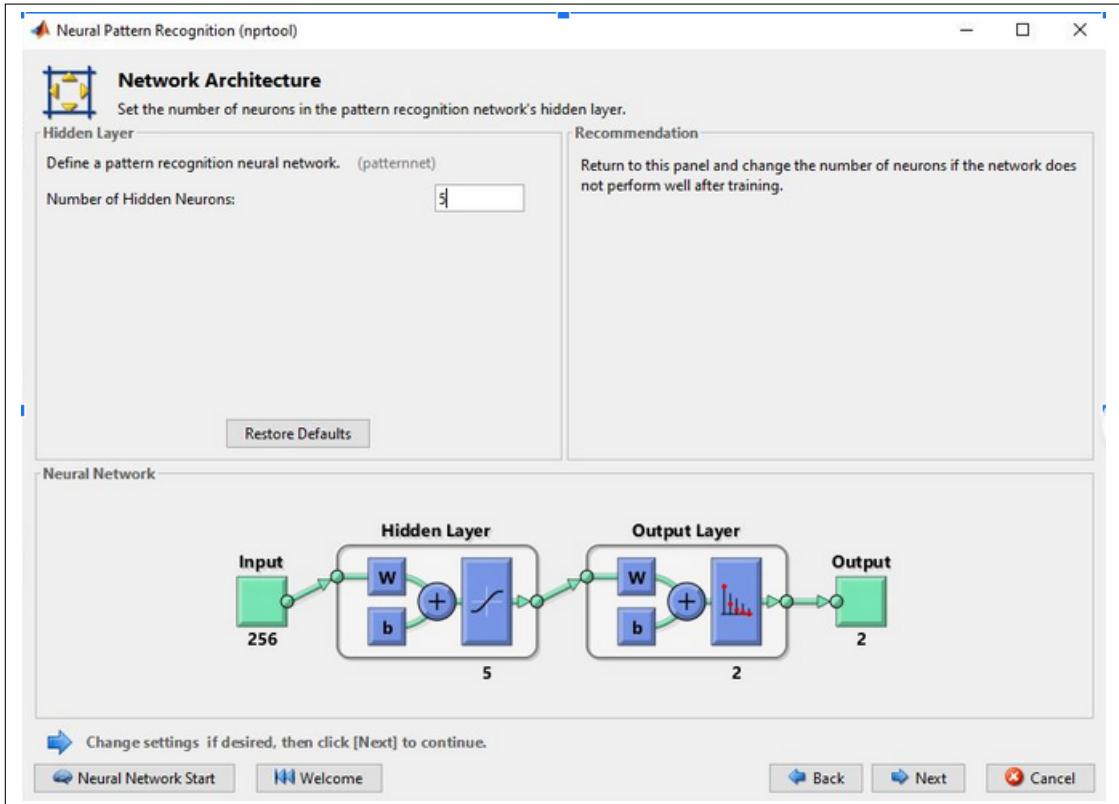
Figura 40 – NFTool - Selecionando Dados.



Fonte – Autor, 2019.

Na figura 41 é necessário definir quantos neurônios serão utilizados para o treinamento. Sendo que, quanto mais complexa é a rede neural, maior deve ser a quantidade de neurônios.

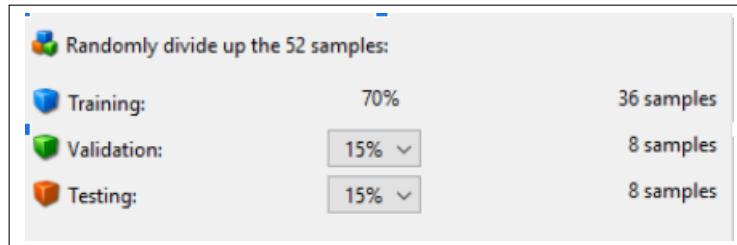
Figura 41 – NFTool - Quantidade de neurônios inseridos para teste.



Fonte – Autor, 2019.

No passo seguinte definimos a quantidade de exemplos para serem treinados, validados e testados do total geral de imagens passadas, sendo a seleção randomicamente como mostra na figura 42.

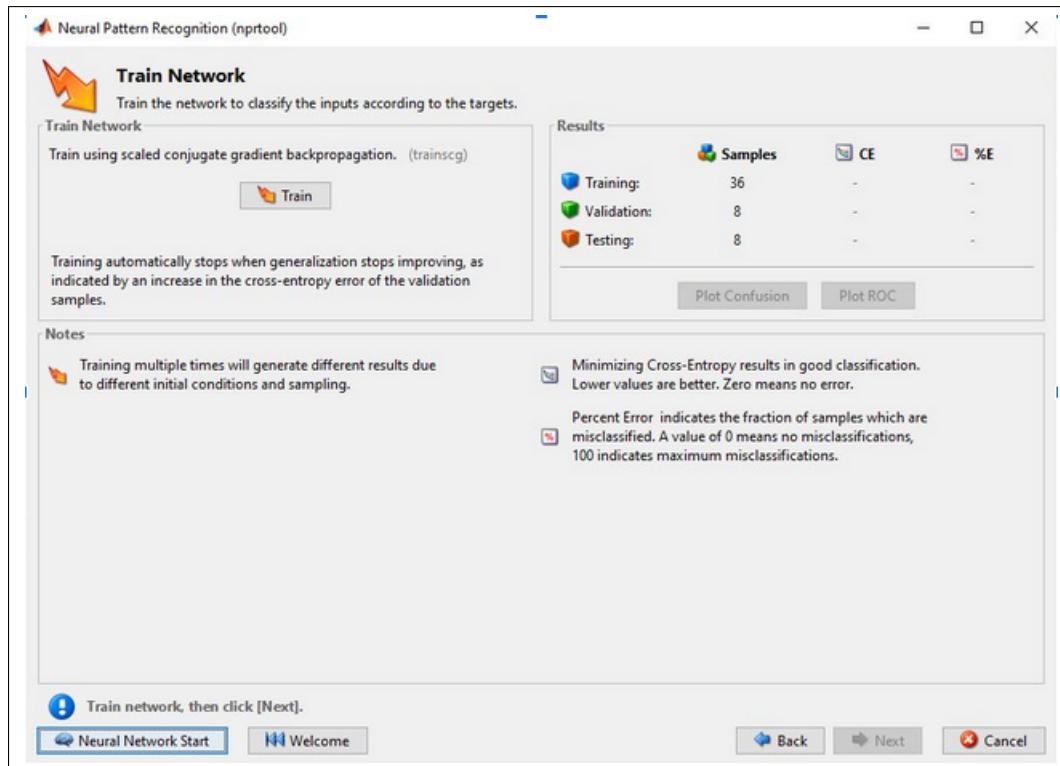
Figura 42 – NFTool - Parâmetros de treinamento.



Fonte – Autor, 2019.

Depois de definir o processo de treino, no próximo passo iremos realizar o treinamento, basta clicar em *Train* como mostra a figura 43.

Figura 43 – NFTool - Etapa de treinamento.

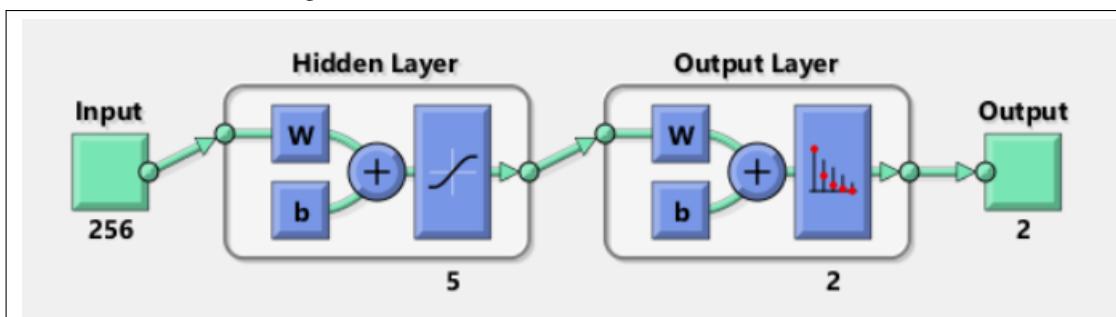


Fonte – Autor, 2019.

4.4.4 Primeiro Treinamento

No primeiro momento definimos as características 5 neurônios para camada oculta como mostra na figura 44.

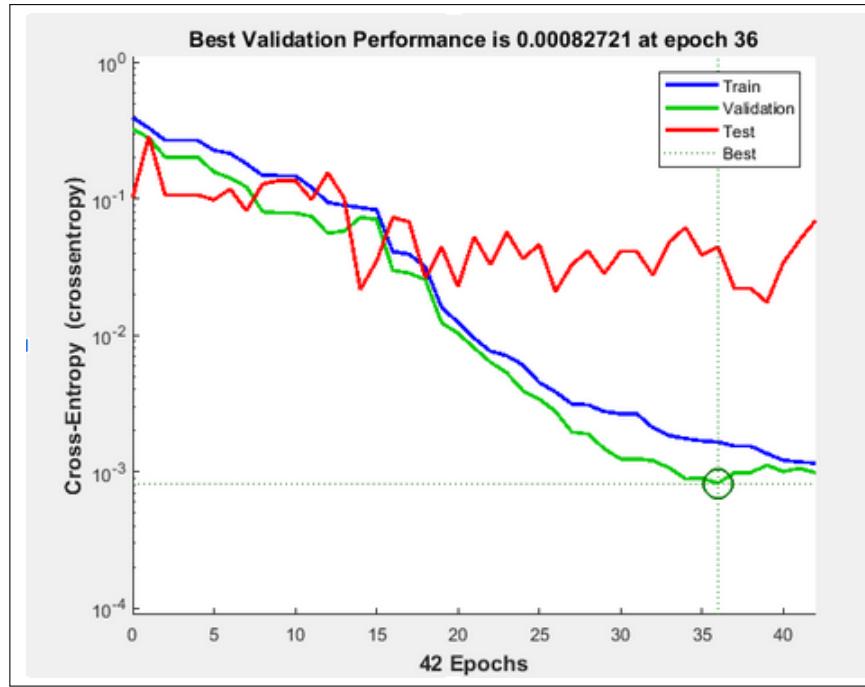
Figura 44 – Primeiro teste de RNA com 5 neurônios.



Fonte – Autor, 2019.

Gerando o treino da RNA através dos dados extraídos dos termogramas de pessoas vivas e em óbito, obteve-se os seguintes resultados, como mostra o gráfico da figura 45, o índice de testes não teve um resultado aceitável pois ficou bem mais elevado que os índices de validação gerando o status de *Underfitting* por falta de treinamentos ou poucos neurônios utilizados.

Figura 45 – Curvas de aprendizado do primeiro teste de RNA.

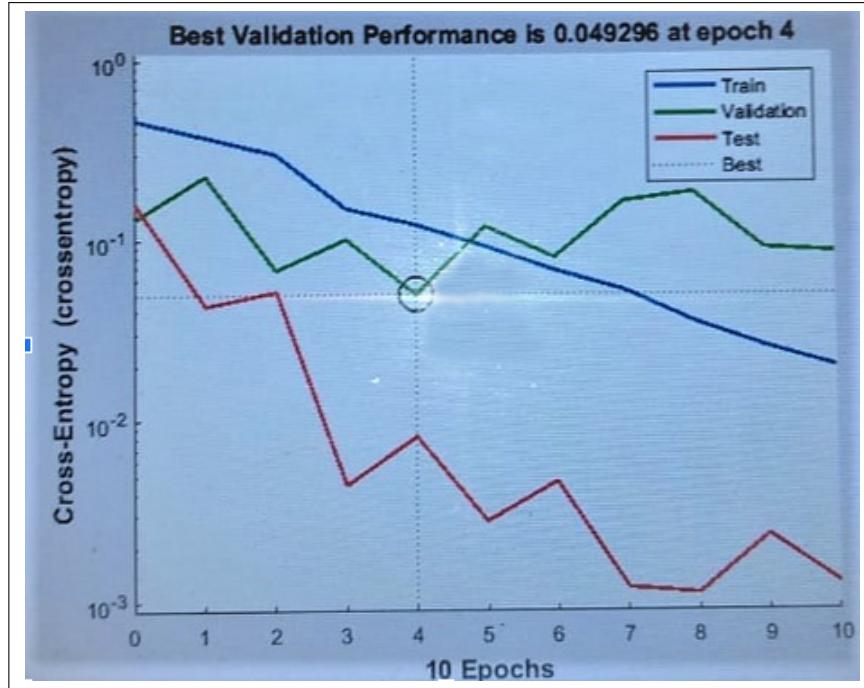


Fonte – Autor, 2019.

4.4.5 Segundo Treinamento

Sendo o primeiro treino não tendo gerado o resultado esperado foi definido 50 neurônios para a camada oculta, e ao obter os resultados a rede se tornou muito especialista ou *status* de *overfitting*, neste caso a rede se torna especialista e se ajusta muito bem ao conjunto de dados anteriormente observado, se mostrando ineficaz ao prever novos resultados, podemos observar pelo segmento de testes onde se apresenta inferior ao segmento de validação representado na figura 46.

Figura 46 – Curvas de aprendizado do segundo teste de RNA.

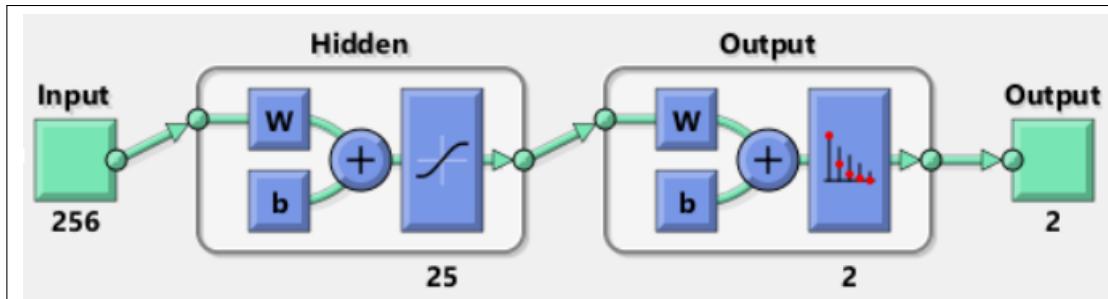


Fonte – Autor, 2019.

4.4.6 Terceiro Treinamento

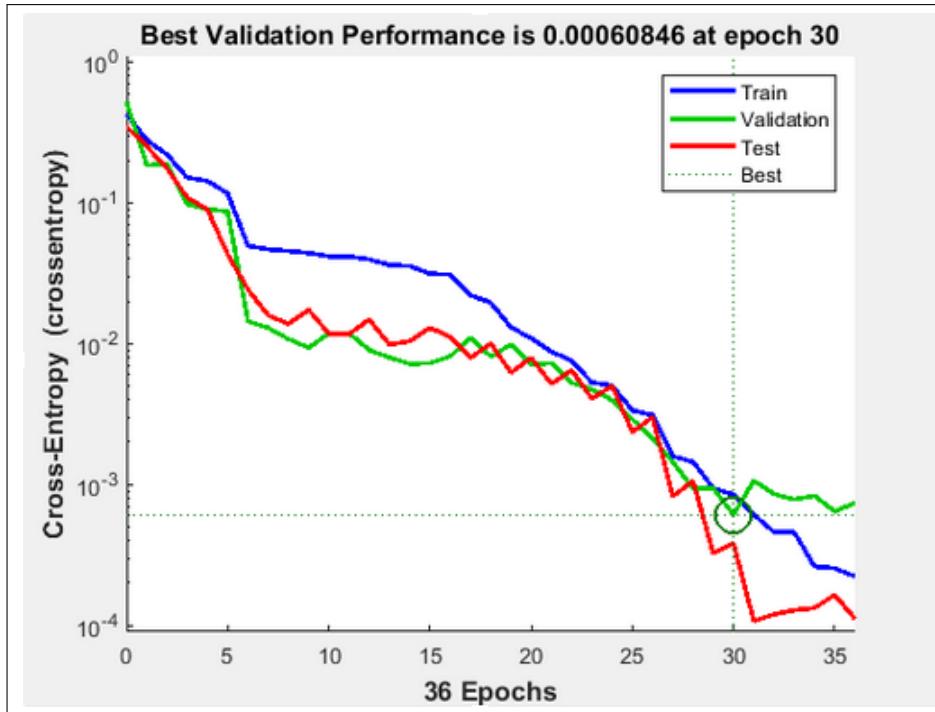
Sendo o segundo treino sem o resultado aceitável da RNA foi parametrizado 25 neurônios para a camada oculta, e ao retreinar a rede obteve-se o resultado abaixo sendo aceitável, podemos perceber que o segmento de *Test* ficou bem aproximado ao segmento de validação representado na figura 48 juntamente com o diagrama da rede representado na figura 47.

Figura 47 – Primeiro teste de RNA com 25 neurônios.



Fonte – Autor, 2019.

Figura 48 – Curvas de aprendizado do terceiro teste de RNA.

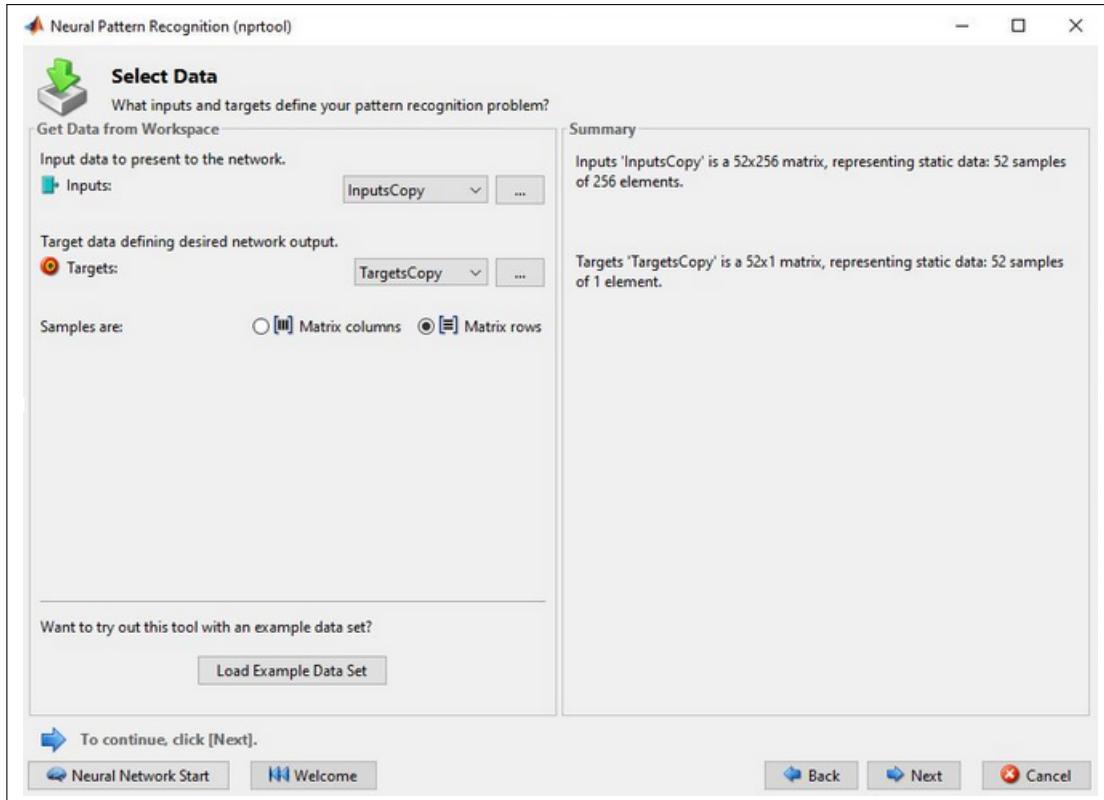


Fonte – Autor, 2019.

4.4.7 Quarto Treinamento

O resultado do terceiro treino se demonstra capaz de tratar o objetivo do final de se definir se a pessoa está viva ou não, mas foi visto a oportunidade de melhorá la, então a rede foi re-treinada sendo as anteriores passando os dados na vertical mudamos para horizontal representado na figura 49 sendo que cada linha agora passou a conter os dados do termograma de um termograma. Outra modificação foi a saída obtida sendo nos treinamentos anteriores foram passados duas saídas e no treino atual apenas uma.

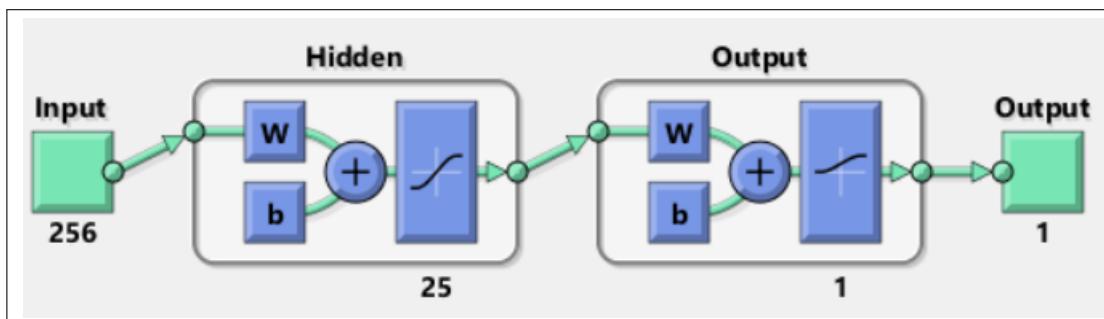
Figura 49 – Tela de entrada dos dados para treinamento.



Fonte – Autor, 2019.

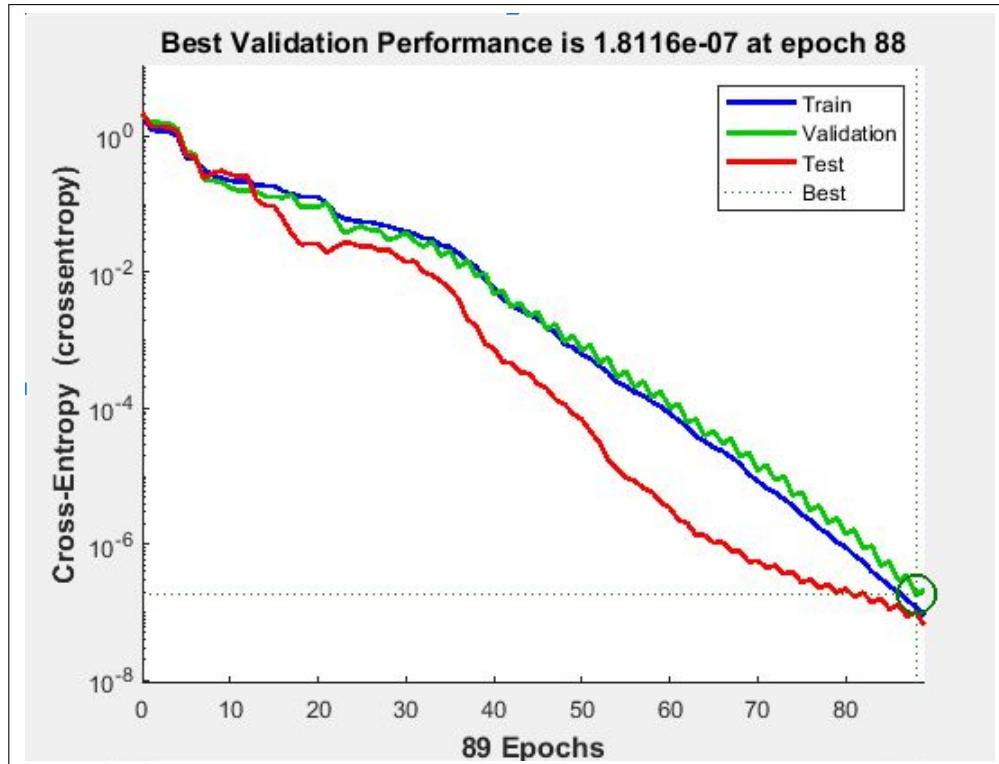
A quantidade de especificações se manteve a mesma sendo 70% dos dados para treino, 15% para validação e 15% para testes com 25 neurônios na camada oculta, obtendo-se o resultado representado na figura 51 onde demonstra uma melhoria no reconhecimento da rede, e na figura 50 representa o diagrama na rede final criada.

Figura 50 – Primeiro teste de RNA com 25 neurônios.



Fonte – Autor, 2019.

Figura 51 – Curvas de aprendizado do quarto teste de RNA.

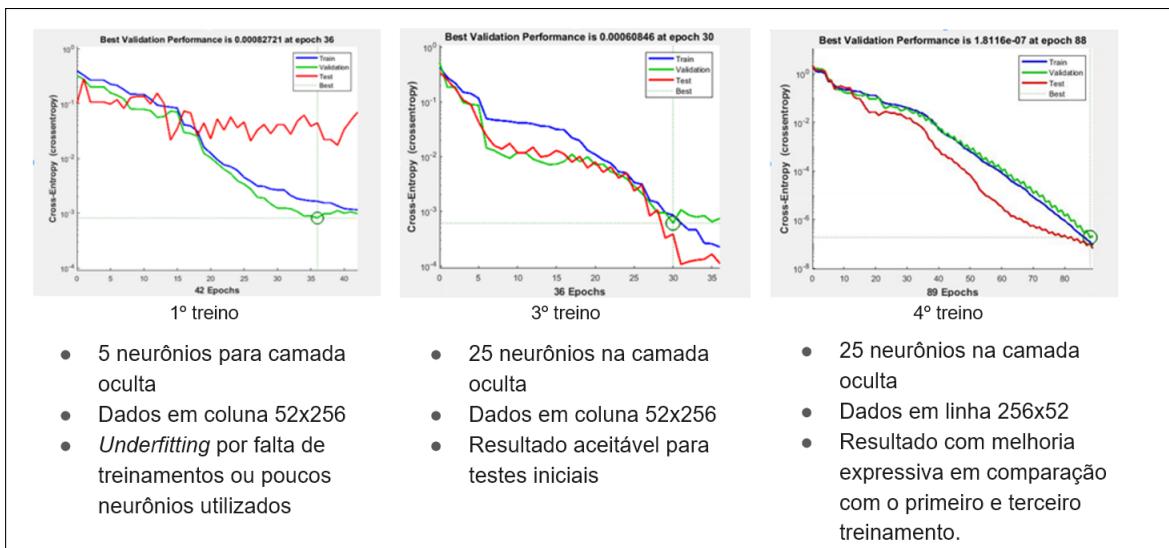


Fonte – Autor, 2019.

4.4.8 Comparação dos treinamentos

A figura 52 representa a comparação do desenvolvimento da RNA em suas etapas.

Figura 52 – Comparação entre os treinamentos realizados.



Fonte – Autor, 2019.

4.4.9 Testando a RNA no MATLAB

Após a rede salva testamos a rede com imagens não treinadas para avaliar o seu funcionamento. Para realizar o teste é necessário extrair a rede do arquivo “*Results*” gerado pelo nptool. esse processo é feito na janela de comando do MatLab com o comando “*net=results.net*”. É necessário também gerar os arquivos dos termogramas a serem gerados um com *status* de vida positivo e a outra negativa, lembrando também que a entrada da imagem deve ser feita na vertical apesar da nova rede ser treinada na horizontal, então teremos os seguintes arquivos no *workspace* do Matlab representados em azul na figura 53.

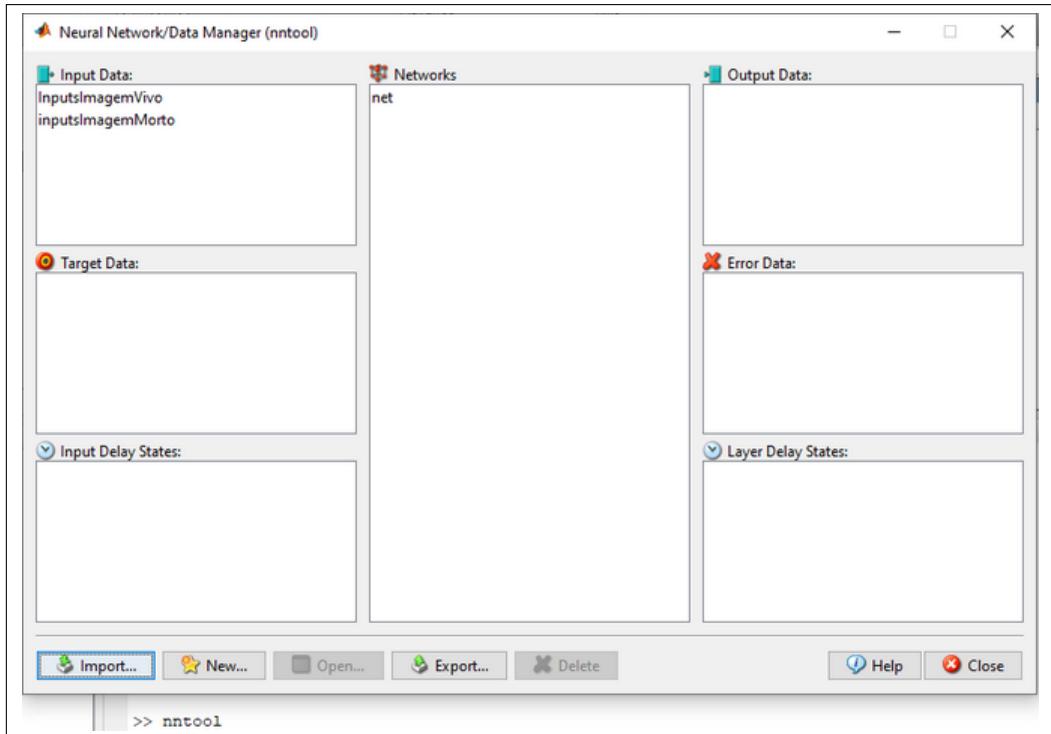
Figura 53 – *Workspace* MatLab.

Name	Value
Inputs	256x52 double
InputsCopy	52x256 double
inputslImagenMorto	256x1 double
inputslImagenVivo	256x1 double
net	1x1 network

Fonte – Autor, 2019.

Em seguida após preparamos os arquivos para teste, inserimos o comando *nntool* na janela de comando do MatLab para testar a rede, então importamos os arquivos para realizar o teste como mostra na figura 54.

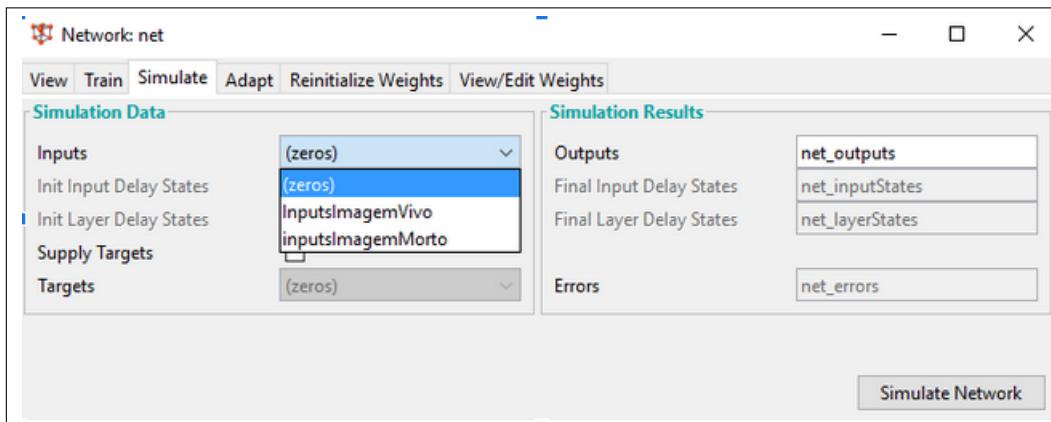
Figura 54 – Ambiente de testes do *nntool*.



Fonte – Autor, 2019.

Tendo importados os arquivos basta dar dois clique na rede onde abrirá a janela representada na figura 55, então selecionamos a aba “*Simulate*” e no campo *inputs* passamos os arquivos a serem testados e depois clicamos em *Simulate Network*.

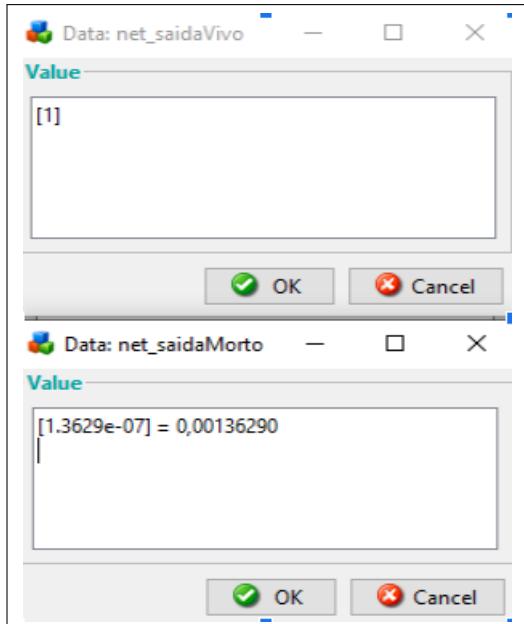
Figura 55 – Seleção na ferramenta *nntool*.



Fonte – Autor, 2019.

Sendo simulado os termogramas passados a obteve-se os resultados representados na figura 56 onde para o termograma de uma pessoa viva o resultado deve ser o mais aproximado de 1 ou o próprio 1 e para o termograma da imagem de uma pessoa morta o mais aproximado de 0 ou o próprio 0. Concluindo então que a rede é capaz de reconhecer uma pessoa de *status* vivo ou morto.

Figura 56 – Resultados obtidos da RNA.



Fonte – Autor, 2019.

4.4.10 Salvando RNA

O passo seguinte é salvar a RNA criada ou melhor precisamos extrair os pesos da rede para ser implementada em python. Para isso utilizamos os comandos representados na figura 57, esses arquivos irão ser gerados no *workspace* do Matlab representados em azul na figura 58, após essa etapa podemos copiar com mais facilidade os dados da rede que foi criada.

Figura 57 – Comandos para extrair dados da RNA treinada.

```
IW=net.IW{1}
LW=net.LW{2}
bIW=net.b{1}
bLW=net.b{2}
MinMax_entrada=net.inputs{1}.range
```

Fonte – Autor, 2019.

Figura 58 – *Workspace* matlab com os dados extraídos da rede.

Name	Value
blW	25x1 double
blLW	-0.1934
Inputs	256x52 double
InputsCopyCopy	52x256 double
inputslImagenMorto	256x1 double
inputslImagenVivo	256x1 double
IW	25x256 double
LW	1x25 double
MinMax_entrada	256x2 double

Fonte – Autor, 2019.

4.5 Codificação

4.5.1 Introdução

O capítulo apresenta um breve reteiro descritivo da codificação do sistema. Será detalhado alguns pontos como:

- Obtenção da RNA: tópico explicativo sobre a ferrenta utilziada para obter a RNA treinada, de forma que entregue ao Python e ele possa utilizar;

4.5.2 Obtendo a RNA

A rede pode ser obtida no site da Darknet(<https://pjreddie.com/darknet/yolo/>) em diferentes versões open-source, então obtemos os seguintes arquivos “yolov3.weights”, “yolov3.cfg” e “class.names”, sendo respectivamente o classificador, as configurações do classificador e as classes de objetos que a rede reconhece.

Os arquivos foram colocados um em cada pasta assim gerando um padrão de organização representado na figura 59.

Figura 59 – *Workspace* Algoritmo.

Nome
cfg
classifier
data

Fonte – Autor, 2019.

Após termos obtido a rede pré treinada começamos a codificar onde o primeiro passo é importar as bibliotecas a serem usadas .Figura 60.

Figura 60 – Importando Bibliotecas no Python.

```

1 #Importando bibliotecas
2 from matplotlib import pyplot as plt
3 import numpy as np
4 import string as s
5 import cv2
6 import csv

```

Fonte – Autor, 2019.

Depois é preciso carregar a rede pré-treinada da YOLO que obtemos, utilizamos a uma função do OpenCV para carregar a rede “cv2.dnn.readNet” essa função permite criar e manipular redes neurais artificiais abrangentes representado na figura 61.

Figura 61 – Carregando RNA por meio do YOLO dentro do Python.

```

9 net = cv2.dnn.readNet("classifier/yolov3.weights", "cfg/yolov3.cfg")
10 #Carregando as classes do classificador
11 classes = []
12 with open("data/class.names", "r") as f:
13     classes = [line.strip() for line in f.readlines()]

```

Fonte – Autor, 2019.

Uma desvantagem de utilizar o OpenCV é que só funciona com CPU, e para processamento em tempo real para processar vídeos seria ideal então utilizar o Darknet criado pelo desenvolvedor da YOLO que também tem sua desvantagem ao ser específico para sistemas linux ou o Darkflow é a adaptação do darknet ao Tensorflow que por sua vez tem a desvantagem de ser muito complexo para instalar mais diferente do Darknet é compatível com Linux, Windows e Mac.

Como o processamento será apenas em imagens por não termos o hardware da câmera, optamos por utilizar o OpenCV que tem como vantagem funcionar sem precisar instalar nada, exceto opencv.

Python
<https://pysource.com/2019/06/27/yolo-object-detection-using-opencv-with-python/>
by Sergio Canu June 27, 2019

Em seguida, carregamos as imagens onde será aplicado a detecção de objetos, passando duas imagens sendo uma digital exibindo o espectro visível ao olho humano e a outra

o termograma da primeira imagem, pegamos o tamanho da primeira imagem para garantirmos que a segunda imagem tenha o mesmo tamanho.

Nessa etapa passamos duas imagem para simular a câmera termográfica onde a mesma enxerga os dois campos de visão trabalhados representado na figura 62.

Figura 62 – Carregando Imagens.

```

22 # Carregando imagens para teste
23 img = cv2.imread("m1.jpg")#imagem cam 1 Digital
24 imgTermica= cv2.imread("m2.jpg")#imagem cam 2 Termograma
25
26 ##Garantindo que as duas imagens tenham o mesmo tamanho para demonstração da imagem sobreposta.
27 height, width, channels = img.shape#Pega os valores do tamanho da imagem
28 imgTermica=cv2.resize(imgTermica,(width,height))

```

Fonte – Autor, 2019.

O próximo passo é passar a imagem digital do espectro visível ao olho humano para ser classificado, nesse caso o classificador do YOLO aceita 3 tamanhos sendo eles :

- 320 320 é pequeno, portanto, menos precisão, mas melhor velocidade;
- 609 609 é maior, portanto, alta precisão e velocidade lenta;
- 416 x 416 está no meio e você obtém um pouco dos dois.

Foi utilizado a função Blob para extrair os recursos da imagem e redimensionar representado na figura 63 e após redimensionar é passado para a entrada da rede e é obtido a saída em formato de matriz com todas as informações dos objetos ,sua posição e acuracidade.

Figura 63 – Redimencionando a imagem.

```

39 blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
40
41 net.setInput(blob)
42 outs = net.forward(output_layers)

```

Fonte – Autor, 2019.

Após obter a Saída a detecção está concluída então é utilizado o código abaixo para extrair apenas os objetos com acurácia acima de 0.5, caso abaixo é ignorado. É tratado também os boxes gerados a mais para cada objeto então é utilizada a função “cv2.dnn.NMBoxes()” para eliminar o ruído dessas caixas Chamamos de Supressão Máxima representado na figura 64.

Figura 64 – Classificação dos dados

```

45 # estrutura dos objetos encontrados
46 class_ids = []#classes
47 confidences = []#acuracidade
48 boxes = []#localização
49 ▼ for out in outs:
50     for detection in out:
51         scores = detection[5:]
52         class_id = np.argmax(scores)
53         confidence = scores[class_id]
54         if confidence > 0.5 :
55             # Object detected
56             center_x = int(detection[0] * width)
57             center_y = int(detection[1] * height)
58             w = int(detection[2] * width)
59             h = int(detection[3] * height)
60             # Rectangle coordinates
61             x = int(center_x - w / 2)
62             y = int(center_y - h / 2)
63
64             boxes.append([x, y, w, h])
65             confidences.append(float(confidence))
66             class_ids.append(class_id)
67
68 indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

```

Fonte – Autor, 2019.

4.5.3 Separando objetos a serem trabalhados

No próximo passo separamos somente os objetos da classe “person” representado na figura 65 onde pegamos o frame da imagem onde o objetos person foi detectado e extraímos os dados do frame para chamada da RNA que criamos no MatLab exemplificando na figura 66.

Figura 65 – Exemplificando frame a ser trabalhado na segunda RNA.



Fonte – Autor, 2019.

Imagen gerada através do código funcionando (imagem a esquerda demonstra o campo visível ao olho humano e a da direita o termograma da imagem gerado por uma câmera termográfica).

Figura 66 – Chamada da função da segunda RNA.

```

74 def classifierObjectRectangle(indexes,img,imgTermica,height, width, channels,class_ids,confidences):
75     font = cv2.FONT_HERSHEY_DUPLEX
76     for i in range(len(boxes)):
77         if i in indexes:
78             x, y, w, h = boxes[i]
79             label = str(classes[class_ids[i]])
80             acuracia= str(round((confidences[i]),2)*100)+"%"
81             if label.strip() == 'person':
82                 label=label+acuracia
83                 color= colors[i]
84
85             #Crop imagem infra
86             imgNova= imgTermica[y:y+h, x:x+w]# variavel a ser utilizada com a nova imagem
87             histo_Termico = plt.hist(imgNova.ravel(),256,[0,256])
88             labelStatus= classifierObjectStatus(histo_Termico)#passar o histograma da imagem e obtem o retorno

```

Fonte – Autor, 2019.

4.5.4 Utilizando a rede gerada no MatLab

No passo seguinte representado na figura 67 na linha [IN]: 88 é realizada a chamada da função “*classifierObjectStatus*” onde é passado o histograma da região do termograma onde foi identificado uma pessoa, onde na figura 67 demonstra a tratativa dos dados e depois é chamado a classe da RNA passando os dados tratados que após receber o retorno da RNA foi definido como acurácia de que valor retornado maior que 0.5 pertence a classe de pessoas vivas e menores que 0.5 a classe de pessoas mortas.

Figura 67 – Função de classificação do retorno da segunda RNA.

```

109 def classifierObjectStatus(histo_Termico):
110
111     for i in range(0,256):
112         entrada[i][0]=histo_Termico[0][i]
113
114     res=netStatus.myNeuralNetworkFunction(entrada)
115
116     if res == -1:
117         return "Não Classificado"
118     if res>0.5:
119         return str(round(res*100,2)) +"% Vítima Viva"
120     else:
121         return str(round(res*100,2)) +"% Vítima Morta"
122

```

Fonte – Autor, 2019.

Na classe Network representado na figura 68 foi atribuído às variáveis constantes os pesos extraídos da RNA criada do MatLab, esses pesos serão utilizados pela rede para retornar o *status* de vida da pessoa.

Figura 68 – Classe Network

```

class Network:
    def myNeuralNetworkFunction(entrada):
        ===== NEURAL NETWORK CONSTANTS =====
        minp = [2901,757,511,121,69,47,59,35,35,36,27,24,38,35,22,28,1
        #minp=-1
        maxp = [241807,56989,69126,58209,59937,71081,82276,89427,91186
        # Layer 1
        b1=[[ -1.4358],[-1.4808],[1.3004],[-1.0322],[ 0.94753],[ -0.778
        IW1_1=[

[0.0845323588824469, -0.0580865530079931, -0.101537332925720, 0.0
[0.926137212496469, -0.131852809429602, -0.247543567232859, -0.47
[-0.650921625899600, 0.0597820277519894, 0.152263965369407, 0.43
[-0.00773532100717570, -0.101388089253132, -0.0257478819904718, 0
[-0.0985440800964646, 0.0253107796636898, -0.0480824472374023, 0.
[0.0184397496597338, 0.0975483292216255, 0.00135372790592467, -0.
[-0.0354833751063269, 0.104122096530415, -0.160750662449114, 0.04
[-0.0759296270114709, -0.0620079685166127, 0.170799805799994, 0.2
[-0.721348835651940, -0.0137048992201925, 0.266822358684806, 0.47

```

Fonte – Autor, 2019.

Sendo ativada a classe para classificar o *status* da pessoa detectada, o algoritmo representado na figura 69 determina se os valores de entrada estão dentro dos valores mínimos e máximos da rede, sendo assim algum dado de entrada fora do mínimo e máximo definido pela rede o algoritmo retorna -1 assim não classificando os dados de entrada. Esse processo é necessário para que o algoritmo não classifique de forma errada os dados de entrada.

Sendo os dados de entrada dentro dos valores mínimos e máximos da RNA o algoritmo normaliza os dados de entrada e em seguida começa o processo de classificação.

Figura 69 – Multiplicação da entrada vs pesos.

```

72 ▼      for i in range(0,25):
73          if ((entrada[i][0] < minp[i]) or (entrada[i][0] > maxp[i])):
74              erro = -1
75
76 ▼      if erro==0:
77          for h in range(0,1):
78
79              for i in range(0,(256-1)):# Normaliza dados
80                  y[i][0]=normaliza(minp[i],maxp[i],entrada[i][0])
81
82 ▼          for i in range(0,(25-1)):
83              for j in range(0,(256-1)):
84                  x[i][1]= x[i][1]+ (y[j][0]*IW1_1[i][j]) # Multiplica a entrada vs Layer 1
85
86                  rsB1=x[i][1]+b1[i][0]# soma a Multiplicação de entrada com IW mais 0 bias 1
87                  y[i][1]= tgHiperbolica(rsB1) # Aplica a tangente hiperbolica
88
89              for j in range(0,(25-1)):
90                  x[i][2]= x[i][2]+ (y[j][1]*LW2_1[j]) #Multiplica a entrada vs Layer 2
91
92                  y[i][2]=x[i][2]+b2
93
94                  yNor[i][1]=sigmoid(y[i][2])
95                  result= yNor[i][1]
96
97          return result
98 ▼      else:
99          return -1

```

Fonte – Autor, 2019.

No processo de classificação o algoritmo multiplica os valores de entrada pela camada oculta IW onde a matriz de entrada é de 256x1 e a camada IW é de 25x256, obtendo o resultado da multiplicação uma matriz de 25x1. O resultado obtido é somado ao bias1 de 25x1, o resultado dessa soma é passado para função da tangente hiperbólica representado na figura 70 que retorna o valor da soma numa matriz de 25x1.

Figura 70 – Tangente Hiperbólica

```
108     def tgHiperbolica(z):
109         return 2.0/(1.0+np.exp(-2*z))-1
110
```

Fonte – Autor, 2019.

Após o retorno do cálculo da primeira camada é realizado a multiplicação pela camada LW onde a matriz resultante de 25X1 é multiplicada pela matriz LW 1x25 obtenido o resultado de 1x1 por sua vez é somado ao bias2 de 1x1 ,sendo o resultado da soma passado para a função de ativação sigmoidal representada na figura 71 que por sua vez retorna o resultado da classificação.

Figura 71 – Função de Ativação

```
105     def sigmoid(z):
106         return 1.0/(1.0+np.exp(-z))
107
```

Fonte – Autor, 2019.

Sendo a classificação realizada o algoritmo principal recebe o valor da classificação e sendo status -1 indica que a RNA não conseguiu classificar as entradas passadas, sendo o retorno com acurácia maior de 0.5 classifica como uma pessoa viva , e menor de 0.5 classifica como uma pessoa morta como representado na figura 73 a verificação de qual classe o retorno pertence e na figura 72 a classificação realizada.

Figura 72 – Retorno da Classificação

```
116     if res == -1:
117         return "Não Classificado"
118     if res>0.5:
119         return str(round(res*100,2)) +"% Vítima Viva"
120     else:
121         return str(round(res*100,2)) +"% Vítima Morta"
```

Fonte – Autor, 2019.

Figura 73 – Retorno da Classificação exibidos na Imagem



Fonte – Autor, 2019.

Como foi utilizado frames de imagens em movimento pode se observar que na segunda classificação da esquerda para direita na figura 73 na visão inicial da imagem o algoritmo reconhece que é uma pessoas com 76% de acurácia e classifica como morta com 1,95% de acurácia devido ao frame do termograma estar um quadro na frente da imagem do campo visível ao olho humano. Com a classificação realizada o algoritmo retorna o resultado representado na figura 74.

Figura 74 – Resultados da classificação referentes a figura 73

Vítimas não classificadas: 0.0
Vítimas vivas: 4.0
Vítimas mortas: 1.0

Fonte – Autor, 2019.

Na figura 73 representa a visão sobreposta da câmera termográfica, sendo assim a figura 75 e 76 demonstra as duas visões da câmera.

Figura 75 – Retorno da Classificação exibidos na Imagem



Fonte – Autor, 2019.

Figura 76 – Retorno da Classificação exibidos na Imagem



Fonte – Autor, 2019.

5 CONSIDERAÇÕES FINAIS

O presente trabalho possibilitou o desenvolvimento de um agente inteligente que utilizando técnicas de aprendizagem de máquina através foi possível reconhecer padrões em uma imagem. O *software* Matlab através da ferramenta *Neural Pattern Recognition* possibilitou modelar a RNA para atingir o objetivo sugerido. Além disso, através do estudo realizado foi possível perceber que através das redes neurais e a IA é possível realizar infinitas aplicações que poderiam ser utilizadas para diversas finalidades.

O estudo aplicado no presente trabalho implica em reconhecer pessoas vivas em ambientes hostis, sendo o objetivo geral concluído se mostrando eficaz na classificação. No entanto, o agente não é capaz de classificar o estado clínico da vítima sendo o um fator crucial para o direcionamento do atendimento pelo agentes externo (socorrista), fator que é possível avaliar apenas pessoalmente. Como trabalho futuro, tem a proposta de, adquirir uma câmera termográfica além de implementar a classificação do estado clínico da vítima melhorando a base de dados da rede com novas imagens adquiridas direto da câmera termográfica.

5.1 Trabalhos futuros

O agente desenvolvido foi capaz de atender os objetivos gerais e específicos, mas além das propostas anteriores para trabalhos futuros pretende-se adaptar-se o agente para trabalhar de forma autônoma juntamente com um drone que possa mapear o ambiente e realizar o reconhecimento retornando a localização da vítima através de um Sistema de Posicionamento Global (GPS), assim dispensando a necessidade de ter uma pessoa operando o sistema, mas analisando somente os dados retornados pelo agente.

REFERÊNCIAS

PYTHON Software Foundation 1. [S.l.: s.n.].

1, P. S. F. *Install Python 3.5.0 (32-bit)*. 2019. Disponível em: <<https://static.filehorse.com/screenshots/developer-tools/python-screenshot-01.png>>. Acesso em: 04 de Outubro de 2019.

2, F. S. P. *Python: Getting started*. 2003. Disponível em: <<https://www.python.org/about/>>. Acesso em: 21 de Junho de 2019.

ACADEMY, D. S. Inteligência artificial. 2018. Disponível em: <<http://datascienceacademy.com.br/blog/o-que-e-visao-computacional/>>. Acesso em: 24 de Outubro de 2019.

ADAMS, F. **The Genuine Works Of Hippocrates**. [S.l.: s.n.], 1939. Acesso em: 26 de Agosto de 2019.

ALBUQUERQUE, M. P. de. **Imagen - Áreas corretas**. cbpf, 2001. Disponível em: <<http://www.cbpf.br/cat/pdsi/visao/>>. Acesso em: 04 de Junho de 2019.

BACKES, A. R.; JUNIOR, J. J. de M. S. **Introdução à Visão Computacional Usando MATLAB**. [S.l.]: Alta Books, 2016. Acesso em: 21 de Outubro de 2019.

BALLARD, D. H.; M., C. **Computer Vision**. [S.l.]: AAAI Press, 2007. Acesso em: 21 de Outubro de 2019.

BISHOP, C. M. Neural networks for pattern recognition. 1995. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.679.1104&rep=rep1&type=pdf>>. Acesso em: 07 de Setembro de 2019.

BOOK, D. L. Funções de ativação. 2017. Disponível em: <<http://deeplearningbook.com.br/funcao-de-ativacao/>>. Acesso em: 04 de Outubro de 2019.

BOOK, D. L. **Neurônio Biológico**. 2018. Disponível em: <<https://i1.wp.com/deeplearningbook.com.br/wp-content/uploads/2018/01/neuronio.jpg?w=800>>. Acesso em: 04 de Outubro de 2019.

BRAGA, A. de P.; LUDEMRIR, T. B.; CARVALHO, A. C. P. de L. F. **Redes Neurais Artificiais**. [S.l.]: LTC, 2000. 5,6,31,49,59 p. Acesso em: 01 de Outubro de 2019.

BRASIL, N. U. Onu: Brasil está entre os 10 países com maior número de afetados por desastres nos últimos 20 anos. Nações Unidas Brasil, 2015. Disponível em: <<https://nacoesunidas.org/onu-brasil-esta-entre-os-10-paises-com-maior-numero-de-afetados-por-desastres-nos-ultimos-20-anos/>>. Acesso em: 24 de Novembro de 2019.

BRIOSCHI, D. M. A história da termografia médica. **Infraredmed**, 2017. Disponível em: <<https://infraredmed.com/2017/04/12/historia-da-termografia-medica/>>. Acesso em: 01 de Maio de 2019.

BROOKSSHEAR, J. G.; SMITH, D. T. **Ciência da Computação**: Uma visão abrangente. Bookman Companhia Editora Ltda, 2013. Disponível em: <<https://integrada.minhabiblioteca.com.br/#/books/9788582600313/cfi/0!/4/4@0.00:41.7>>. Acesso em: 26 de Agosto de 2019.

BV, T. S. TIOBE Index for September 2019: September headline: Php is struggling to keep its top 10 position. 2019. Disponível em: <<https://www.artima.com/intv/python3.html>>. Acesso em: 21 de Junho de 2019.

CARVALHO, L. M. M. de. Bases sobre a teoria da cor aplicada aos sistemas digitais. Escola Dr. Manuel Laranjeira, 2012–2013. Disponível em: <<https://utilizaodosistemamultimidia.weebly.com/bases-sobre-a-teoria-da-cor-aplicada-aos-sistemas-digitais.html>>. Acesso em: 26 de Agosto de 2019.

CATARINA, U. F. de S. Atlas brasileiro de desastres naturais 1991 a 2012: Volume amazonas. CEPED UFSC, 2013. Disponível em: <http://www.ceped.ufsc.br/wp-content/uploads/2012/01/AMAZONAS_mioloWEB.pdf>. Acesso em: 06 de Junho de 2019.

CHEN, L.-C. *et al. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs*. 2014. 62 p. Disponível em: <<https://arxiv.org/abs/1412.7062v4>>. Acesso em: 01 de Setembro de 2019.

COHEN, J. G. **Medical Thermography**: Scientific american february 1967. [S.l.]: Medline, 1967. v. 216. Acesso em: 06 de Junho de 2019.

CONCI, A.; AZEVEDO, E.; LETA, F. R. **Computação Gráfica**: Teoria e prática. [S.l.]: Elsevier, 2009. v. 2. Acesso em: 03 de Outubro de 2019.

CURCIO, B.; HABERMAN, J. **Infrared thermography: a review of current medical application, instrumentation and techniques**: Radiol. technol. [S.l.]: Medline, 1971. v. 42. 233-247 p. Acesso em: 06 de Junho de 2019.

DYE. **The evolution of chiropractic**: Its discovery and development. [S.l.]: A. Aug. Dye, 1939. Acesso em: 06 de Junho de 2019.

EDUCAÇÃO, C. E. Funções e sinais vitais. 2018. Disponível em: <<https://cursos.escolaeducacao.com.br/artigo/funcoes-e-sinais-vitais>>. Acesso em: 26 de Agosto de 2019.

FACURE, M. Funções de ativação: Entendendo a importância da ativação correta nas redes neurais. 2017. Disponível em: <<https://matheusfacure.github.io/2017/07/12/activ-func/>>. Acesso em: 5 de Outubro de 2019.

FALCÃO, A. X. **Apresentação do Curso e da Área de Processamento de Imagem Digital**. UNICAMP, 2014. Disponível em: <<http://www.ic.unicamp.br/~afalcao/mo443/slides-aula1.pdf>>. Acesso em: 04 de Junho de 2019.

FILHO, O. M.; NETO, H. V. **Processamento Digital de Imagens**: Brasport hall. [s.n.], 1999. 9 p. Disponível em: <<http://www.ogemarques.com/wp-content/uploads/2014/11/pdi99.pdf>>. Acesso em: 29 de Agosto de 2019.

FLIR. **O que é Infravermelho?** 2019. Disponível em: <<https://www.flir.com.br/discover/what-is-infrared/>>. Acesso em: 01 de Agosto de 2019.

FLIR. **Refrigerada ou Não Refrigerada ?** 2019. Disponível em: <<https://prod.flir.com.br/discover/rd-science/cooled-or-uncooled/>>. Acesso em: 01 de Agosto de 2019.

GOMES, J. **Histograma de imagem e operações baseadas em histograma**: Histograma básico. GitBook, 2019. Disponível em: <<https://jacksongomesbr.gitbooks.io/computacao-grafica/histograma-e-operacoes-baseadas-em-histograma.html>>. Acesso em: 04 de Junho de 2019.

GONZALES, R. C.; WOODS, R. E. **Processamento Digital de Imagens**. 3. ed. [S.l.]: Person, 2009. 1-11 p. Acesso em: 12 de Outubro de 2019.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 2. ed. Prentice Hall, 1992. 1-45 p. Disponível em: <http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/Digital_Image_Processing_2ndEd.pdf>. Acesso em: 28 de Agosto de 2019.

HAYKIN, S. **Redes Neurais**. 2. ed. [S.l.]: bookman, 2003. 174,271-272 p. Acesso em: 01 de Outubro de 2019.

LACERDA, L. H. G.; RESENDE, R. R. Enxergando até no escuro! lente de contato com grafeno permitirá ter visão de calor. **NanoCell News**, v. 1, 2014. Disponível em: <<http://www.nanocell.org.br/enxergando-ate-no-escuro-lente-de-contato-com-grafeno-permitira-ter-visao-de-calor/>>. Acesso em: 06 de Junho de 2019.

LIVRE, M. **Câmera Dji Zenmuse Xt2 Térmica Inspeção Para Drone Matrice**. 2019. Disponível em: <encurtador.com.br/gwxOS>. Acesso em: 04 de Abril de 2019.

LOMAX, E. **Historical development of concepts of thermoregulation**: Body temperature . modern pharmacology . toxicology. New York: Marcel Dekker, 1979. v. 6. Acesso em: 06 de Junho de 2019.

LÓPEZ, C. T. **El espectro electromagnético**. Cuaderno de Cultura Científica, 2016. Disponível em: <<https://culturcientifica.com/2016/08/16/el-espectro-electromagnetico/>>. Acesso em: 04 de Junho de 2019.

MATHWORKS, T. **Adaptive Histogram Equalization**. 2019. Disponível em: <<https://www.mathworks.com/help/images/adaptive-histogram-equalization.html>>. Acesso em: 04 de Junho de 2019.

MATHWORKS, T. **Documentation**: Getting started with matlab. 2019. Disponível em: <https://www.mathworks.com/help/matlab/getting-started-with-matlab.html?s_tid=CRUX_lftnav>. Acesso em: 21 de Agosto de 2019.

MECÂNICO, L. do. **Camera Termica ON PRO USB-C para Smartphone Android FLIR**. 2019. Disponível em: <encurtador.com.br/bCL15>. Acesso em: 04 de Abril de 2019.

MENESES, P. R. et al. **INTRODUÇÃO AO PROCESSAMENTO DE IMAGENS DE-SENSORIAMENTO REMOTO**. CNPq, 2012. 87 p. Disponível em: <<http://memoria.cnpq.br/documents/10157/56b578c4-0fd5-4b9f-b82a-e9693e4f69d8>>. Acesso em: 30 de Agosto de 2019.

NAKAMURA, W. T. **Backpropagation**. 2013. Disponível em: <https://www.researchgate.net/figure/Model-of-the-Backpropagation-Algorithm_fig2_272723714>. Acesso em: 07 de Outubro de 2019.

NOVICK, N. **The VNT photo-eletric instrument. J. Clin. Chiro.** [S.l.]: J. Clin. Chiro (JCC), 1969. v. 2. 78-83 p. Acesso em: 06 de Junho de 2019.

OLIVER, M. 5 tragédias que marcaram o brasil nós últimos tempos. **Nome da Revista**, 2015. Disponível em: <<https://fatosdesconhecidos.ig.com.br/5-tragedias-que-marcaram-o-brasil-nos-ultimos-tempo/>>. Acesso em: 01 de Junho de 2019.

OPENCV. About. 2019. Disponível em: <<https://opencv.org/about/>>. Acesso em: 21 de Junho de 2019.

PIERRO, B. de. Ciência do desastre: Estudo avalia a produção científica sobre catástrofes naturais no mundo e destaca a brasileira na área de hidrologia. Revista Pesquisa FAPESP, 2018. Disponível em: <<https://revistapesquisa.fapesp.br/2018/02/15/ciencia-do-desastre/>>. Acesso em: 06 de Junho de 2019.

PUTLEY, E. **The development of thermal imaging systems.** Plenum Press, 1982. 151-166 p. Disponível em: <https://link.springer.com/chapter/10.1007/978-1-4684-7697-2_21>. Acesso em: 06 de Junho de 2019.

RASK, M. **Thermography and the human spine.** [S.l.]: ORTHOP, 1979. v. 8. 73-82 p. Acesso em: 06 de Junho de 2019.

REDMON *et al.* Yolov3: An incremental improvement. **arXiv**, 2018. Disponível em: <<https://pjreddie.com/darknet/yolo/>>. Acesso em: 21 de Junho de 2019.

REDMON, J. **YOLO: Real Time Object Detection:** How it works. GitHub, 2019. Disponível em: <<https://github.com/pjreddie/darknet/wiki/YOLO:-Real-Time-Object-Detection>>. Acesso em: 04 de Junho de 2019.

REDMON, J. *et al.* You only look once: Unified, real-time object detection: Computer vision and pattern recognition (cs.cv). arxiv.org, 2015. Disponível em: <<https://arxiv.org/abs/1506.02640>>. Acesso em: 10 de Junho de 2019.

RYAN, J. Thermography: Australasian radiology. v. 13, p. 23–26, 1969. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1440-1673.1969.tb01210.x>>. Acesso em: 26 de Agosto de 2019.

SAKURAI, R. **Rede Neural Convolucional.** 2017. Disponível em: <<http://rafaelsakurai.github.io/images/posts/2017-12-20-cnn-mapreduce.png>>. Acesso em: 07 de Outubro de 2019.

SALMERON, R. A. **Eletricidade e Magnetismo (Básico).** [s.n.], 2008. Disponível em: <<http://efisica.if.usp.br/eletrostatica/basico/fenomenos/tipos/>>. Acesso em: 31 de Agosto de 2019.

SHOPTIME. **Câmera Térmica Sensor Térmico Drone Radiométrico Duo R Flir Duo R Para Drones.** 2019. Disponível em: <encurtador.com.br/cnpyD>. Acesso em: 04 de Abril de 2019.

SILVA, I. N. da; SPATTI, R. A. F. D. H. **Redes Neurais Artificiais:** Fundamentos teóricos e aspectos práticos. 2. ed. [S.l.]: Artliber, 2016. 94-100 p. Acesso em: 01 de Outubro de 2019.

SOUZA, J. A. D. Reconhecimento de padrões usando indexação recursiva. 1999. Disponível em: <<https://repositorio.ufsc.br/xmlui/handle/123456789/80484>>. Acesso em: 07 de Setembro de 2019.

SZELISKI, R. **Visão Computacional:** Algoritmo aplicado. [S.l.]: Draft, 2010. Acesso em: 21 de Outubro de 2019.

UFSC. **Perceptron.** 2011. Disponível em: <https://www.gsigma.ufsc.br/~popov/aulas/rna/neuronio_artificial/neuronio_artificial.jpg>. Acesso em: 04 de Outubro de 2019.

V, A. S. Understanding activation functions in neural networks. 2017. Disponível em: <<https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>>. Acesso em: 5 de Outubro de 2019

VENNERS, B. **The Making of Python 1:** A conversation with guido van rossum part i. Artima, 2003. Disponível em: <<https://www.artima.com/intv/python3.html>>. Acesso em: 21 de Junho de 2019.

YOUNG, H. D.; FREEDMAN, R. A. Física iii eletromagnetismo. Pearson Education do Brasil, v. 12, 2009. Disponível em: <https://www.academia.edu/28596962/F%C3%8DSICA_III_ELETROMAGNETISMO_12_a_EDI%C3%87%C3%83O>. Acesso em: 31 de Agosto de 2019.

ZANOTTA, D. C.; FERREIRA, M. P.; ZORTEA, M. **Processamento de imagens de satélite.** Oficina de Textos, 2019. Disponível em: <<https://books.google.com.br/books?id=l6SWDwAAQBAJ&printsec=frontcover&hl=pt-PT#v=onepage&q&f=false>>. Acesso em: 01 de Setembro de 2019.

ANEXO A – CÓDIGO PYTHON RNA YOLO

```

1 from matplotlib import pyplot as plt
2 from Networkstatus import Network as netStatusas
3 import numpy as np
4 import string as s
5 import cv2
6 import csv
7
8 cont=np.zeros((1,4), dtype=np.float64)
9 entrada=np.zeros((256,1), dtype=np.float64)
10 cons=None
11 # Carregando YOLO pela classe dnn.readNet que permite criar e
12 # manipular redes neurais artificiais abrangentes.
13 net = cv2.dnn.readNet("classifier/yolov3.weights", "cfg/yolov3.cfg"
14 )
15 #Carregando as classes do classificador
16 classes = []
17 with open("data/class.names", "r") as f:
18     classes = [line.strip() for line in f.readlines()]
19
20 #Realiza a captura dos nomes das camadas da rede
21 layer_names = net.getLayerNames()
22 output_layers = [layer_names[i[0] - 1] for i in net.
23     getUnconnectedOutLayers()]
24 #define uma cor para cada classe de objetos
25 colors = np.random.uniform(0, 255, size=(len(classes), 3))
26
27 # Carregando imagens para teste
28 img = cv2.imread("imgTest/m1.jpg")#imagem cam 1 Digital
29 imgTermica= cv2.imread("imgTest/m2.jpg")#imagem cam 2 Termograma
30
31 #Garantindo que as duas imagens tenham o mesmo tamanho para
32 #demostra{\c c}\{^a}o da imagem sobreposta.
33 height, width, channels = img.shape#Pega os valores do tamanho da
34 # imagem
35 imgTermica=cv2.resize(imgTermica,(width,height))
36
37 imgDupla = cv2.addWeighted(imgTermica, 0.5, img, 0.5, 0)#cria
38 # imagem sobreposta
39 # estrutura dos objetos encontrados
40 class_ids = []#classes
41 confidences = []#acuracidade

```

```

36 boxes = [] #localizacao
37
38 def classifierObject(img,imgTermica,height , width , channels ,
39   class_ids,confidences,boxes):
40
41   blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0,
42   0), True, crop=False)
43
44   net.setInput(blob)#passando imagem para detectacao na #rede do
45   YOLO
46   outs = net.forward(output_layers)#Saida da rede
47
48   for out in outs:
49     for detection in out:
50       scores = detection[5:]
51       class_id = np.argmax(scores)
52       confidence = scores[class_id]
53       if confidence > 0.5 :
54         # Object detected
55         center_x = int(detection[0] * width)
56         center_y = int(detection[1] * height)
57         w = int(detection[2] * width)
58         h = int(detection[3] * height)
59         # Rectangle coordinates
60         x = int(center_x - w / 2)
61         y = int(center_y - h / 2)
62
63         boxes.append([x, y, w, h])
64         confidences.append(float(confidence))
65         class_ids.append(class_id)
66
67 indexes = cv2.dnn.NMSBoxes(boxes, confidences , 0.5, 0.4)
68 classifierObjectRectangle(indexes,img,imgTermica,height , width ,
69   channels,class_ids,confidences)
70
71 def classifierObjectRectangle(indexes,img,imgTermica,height , width ,
72   channels,class_ids,confidences):
73   font = cv2.FONT_HERSHEY_DUPLEX
74   for i in range(len(boxes)):
75     if i in indexes:
76       x, y, w, h = boxes[i]
77       label = str(classes[class_ids[i]])
78       acuracia= str(round((confidences[i]),2)*100)+"% "
79       if label.strip() == 'person':
80         label=acuracia+label
81         color= colors[i]
82
83
84         #Crop imagem infra
85         imgNova= imgTermica[y:y+h, x:x+w]# variavel a ser
86         utilizada com a nova imagem
87         histo_Termico = plt.hist(imgNova.ravel()
88 ,256,[0,256])
89         labelStatus= classifierObjectStatus(histo_Termico)#
90         pasar o histograma da imagem e obtem o retorno

```

```

85         #cv2.imwrite("person.jpg", imgNova)
86             #nesse trecho de código e contornado o objeto
87             emcontrado.
88             cv2.rectangle(img, (x, y), (x + w, y + h), color,
89             2)
90                 cv2.putText(img, label, (x+5, y + 20), font, 0.5,
91             color, 1)
92                     #imagem infra
93                     cv2.rectangle(imgTermica, (x, y), (x + w, y + h),
94             color, 2)
95                         cv2.putText(imgTermica, labelStatus, (x+5, w*2+10),
96             font, 0.4, color, 1)
97
98
99     resultado = np.zeros((100, 500, 3), dtype=np.uint8)
100    cv2.rectangle(resultado, (0, 0), (500, 100), (255, 255, 255),
101    -1)
102
103    cv2.putText(resultado, "Vitimas nao classificadas: " +str(cont
104 [0][0]),(10, 10*2+10), font,0.8, (0,0,0), 1)
105    cv2.putText(resultado, "Vitimas vivas: " +str(cont[0][1]), (10,
106 25*2+10), font, 0.8, (0,0,0), 1)
107    cv2.putText(resultado, "Vitimas mortas:" +str(cont[0][2]), (10,
108 40*2+10), font, 0.8, (0,0,0), 1)
109    cv2.imshow("Resultado", resultado)
110
111 def classifierObjectStatus(histo_Termico):
112
113     for i in range(0,256):
114         entrada[i][0]=histo_Termico[0][i]
115
116     res=netStatuaas.myNeuralNetworkFunction(entrada)
117
118     if res == -1:
119         cont[0][0]+=1
120         return "Nao Classificado"
121     if res>0.5:
122         cont[0][1]+=1
123         return str(round(res*100,2)) +"% Vitima Viva"
124     else:
125         cont[0][2]+=1
126         return str(round(res*100,2)) +"% Vitima Morta"
127
128
129
130 def main():
131
132     classifierObject(img,imgTermica,height, width, channels,
133     class_ids,confidences,boxes)
134
135     cv2.waitKey(0)
136     cv2.destroyAllWindows()
137
138 if __name__ == "__main__":
139     main()

```

ANEXO B – CÓDIGO PYTHON RNA MLP

```

1  from math import exp
2  import numpy as np
3  from matplotlib import pyplot as plt
4
5
6
7  class Network:
8
9      def myNeuralNetworkFunction(entrada):
10
11          ##### NEURAL NETWORK CONSTANTS #####
12          #minp sao 256 valores
13          minp =
14          [2901,757,511,121,69,47,59,35,35,36,27,24,38,35,22,28,18,.....]
15          #minp=-1
16          #maxp sao 256 valores
17          maxp =
18          [241807,56989,69126,58209,59937,71081,82276,89427,91186,.....]
19
20
21          # Layer 1
22          #b1 sao 25 valores
23          b1=[[ -1.4358],[-1.4808],[1.3004],
24          [-1.0322],[ 0.94753],[-0.77839],
25          [ 0.72987],[-0.55003],[0.63319],
26          [ 0.32434],[-0.23048],[-0.15622],
27          [ 0.0031682],[0.14521],[-0.25386],
28          [ 0.45064],[-0.30156],[-0.59205],
29          [0.47924],[1.0233],[-1.0322],
30          [-1.1708],[1.153],[1.3076],
31          [-1.5085]]
32
33          #IW1_1 sao 25x256
34          IW1_1=[
35          [0.0845323588824469,-0.0580865530079931,-0.101537332925720,
36          0.0232321712902260,-0.0318992900215186,-0.0942764119146778,
37          0.158831216571407,0.0368070314784685,-0.0574976816789267,
38          0.135536571055014,0.138952570669855,0.0895613907032485,
39          -0.0603707980083709,0.0121309210637322,.....]
40
41          # Layer 2

```

```

40     b2=[-0.19337654448585128719]
41     #LW2_1 sao 256 valores
42     LW2_1=[-0.55584628343048991805 ,2.9703894879049075328 ,
43     -2.0549556174115464202 ,-0.6449276227456555155 ,
44     -0.21763611121203940701 ,0.57965006383119543898 ,
45     0.53080728990392167521 ,-0.54466121580492399268 ,
46     -2.183242221425631957 ,.....]
47
48     erro=0
49     result=0
50
51     numneu = [256, 25, 1] #Numero de Neuronios em cada camada
52     #256 entradas
53     #25 neuronios na camada oculta
54     #1 Saída
55
56     #variaveis a serem usadas ja zerando o valor das mesmas
57     y = np.zeros((256,25), dtype=np.float64) #Valores de Entrada
58     256,25
59     yNor = np.zeros((256,25), dtype=np.float64) #Valores
60     Normalizados
61     x = np.zeros((256,25), dtype=np.float64) #Valores de Entrada
62     #yDes = np.zeros((256,25), dtype=np.float64) #Valores
63     Desnormalizados
64
65     ===== SIMULATION =====
66     #Funcao de processamento de entrada minima e maxima do mapa
67
68     for i in range(0,25):
69         if ((entrada[i][0] < minp[i]) or (entrada[i][0] > maxp[i])):
70             erro = -1
71
72     if erro==0:
73         for h in range(0,1):
74
75             for i in range(0,(256-1)):# Normaliza dados
76                 y[i][0]=normaliza(minp[i],maxp[i],entrada[i][0])
77
78             for i in range(0,(25-1)):
79                 for j in range(0,(256-1)):
80                     x[i][1]= x[i][1]+ (y[j][0]*IW1_1[i][j]) # Multiplica a
81                     entrada vs Layer 1
82
83                     rsB1=x[i][1]+b1[i][0]# soma a Multiplicacao de entrada
84                     com IW mais 0 bias 1
85                     y[i][1]= tgHiperbolica(rsB1) # Aplica a tangente
86                     hiperbolica
87
88                     for j in range(0,(25-1)):
89                         x[i][2]= x[i][2]+ (y[j][1]*LW2_1[j]) #Multiplica a
90                     entrada vs Layer 2
91
92                     y[i][2]=x[i][2]+b2
93
94                     yNor[i][1]=sigmoid(y[i][2])
95                     result= yNor[i][1]
96
97
98             return result
99         else:
100             return -1
101
102     def normaliza(minVal,maxVal, value):
103         vn=((2 * (value - minVal)) / (maxVal - minVal))-1
104         return vn
105
106     def sigmoid(z):
107         return 1.0/(1.0+np.exp(-z))
108
109     def tgHiperbolica(z):
110         return 2.0/(1.0+np.exp(-2*z))-1

```