ALGORITMO DE IDENTIFICAÇÃO E EXTRAÇÃO DE TEXTOS EM IMAGENS

Jorge José Fernandes Filho[1],
Adan Lucio Pereira[2]
Graziela Vieira Carneiro[2],
Bruno Bastos Stoll[2]

[1] Discente do curso de Engenharia Elétrica do Centro Universitário MULTIVIX Vitória
[2] Docentes do Centro Universitário MULTIVIX Vitória

RESUMO
A identificação automática de textos em imagens pode gerar ganhos, visando reduzir erros, ganhar em produtividade e diminuir tarefas repetitivas. Nesse contexto, o presente trabalho irá abordar de forma clara e direta um trabalho desenvolvido para identificação de textos utilizando sistemas inteligentes baseados em imagens. A motivação principal é a identificação de objetos, componentes e equipamentos identificados por texto de forma automatizada, em algumas situações que não é possível a presença humana, como em determinadas aplicações industriais em que a ação repetitiva promove erros consideráveis. Para o desenvolvimento deste trabalho foram utilizadas técnicas de sistemas inteligentes como *transfer learning* e tratamento de imagens, o trabalho utilizou da linguagem Python e usou as ferramentas OpenCV para tratar imagens e executar o processamento da rede neural. O objetivo deste trabalho é realizar a identificação de textos em imagens aleatórias, usando o recurso de *transfer learning* de uma rede já treinada e com boa performance.
PALAVRAS-CHAVE
Redes neurais; *transfer learning*; processamento de imagem.

ABSTRACT
Automatic identification of text in images can yield benefits by reducing errors, increasing productivity, and minimizing repetitive tasks. In this context, the present work will clearly and directly address a project developed for text identification using intelligent systems based on images. The main motivation is the automated identification of objects, components, and equipment identified by text in situations where human presence is not possible, such as in certain industrial applications where repetitive actions lead to significant errors. For the development of this work, techniques from intelligent systems such as transfer learning and image processing were utilized. The work was implemented using the Python programming language and employed OpenCV tools for image processing and neural network execution. The objective of this work is to perform text identification in random images, using transfer learning from a pre-trained network with good performance.
KEYWORDS
Neural networks; transfer learning; image processing.

INTRODUÇÃO

A escrita surgiu na pré-história como forma de representar uma informação ou uma ideia (MENDES, 2020). Atualmente usamos a escrita com o mesmo intuito quando representamos um objeto com seu nome, ou quando se escreve um código para cada produto, ou quando há algo escrito em uma placa. Existe atualmente uma necessidade de se automatizar processos visando redução de erros, ganhos de produtividade, diminuição de tarefas repetitivas entre outras vantagens (SCHULTZ,

2020). Nesse sentido existem desafios a serem superados para análise de imagens e extrações de textos, já que a identificação de textos pode gerar ganhos em processos automatizados.

A dificuldade de identificar a localização exata de textos é, de certa forma, desafiadora visto que várias dificuldades são encontradas em imagens reais, tais como deslocamento, inclinação, ruído em imagens, distorções entre outros, dessa forma, algoritmos convencionais têm dificuldades ou exigem grande processamento para localizar e diferenciar uma imagem convencional de uma área onde existe um texto (ROSEBROCK, 2018).

As dificuldades de uso de algoritmos convencionais para a identificação de textos podem ser superadas utilizando tecnologias modernas que vem crescendo muito atualmente, e entre essas tecnologias pode-se destacar as técnicas de *machine learning*, sendo uma dessas técnicas a utilização de *deep learning*, que é uma técnica de redes neurais convolucionais que tem a como característica a capacidade de generalização e aprendizagem (PONTI MA, 2017). As redes neurais são assim conhecidas por ter a capacidade de aprender assim como nosso cérebro, trazendo vantagens visto que se pode treinar essa rede para realizar processamentos e a mesma aprender a executar essas tarefas. Além disso há a características de generalização, podendo ser usados em outros ambientes e imagens nunca antes apresentadas as redes neurais assim como funciona com nosso cérebro.

Um método de identificação de texto viabiliza ferramentas de para uso em indústrias para identificação, seleção ou classificação de objetos, pode-se também desenvolver sistemas inteligentes para auxílio de carros autônomos auxiliando a identificação e leitura de textos em placas em tempo real, outras possibilidades são para classificação de objetos em uma esteira transportadora que separa e classifica as diversas embalagens ou pacotes.

1. REFERENCIAL TEÓRICO

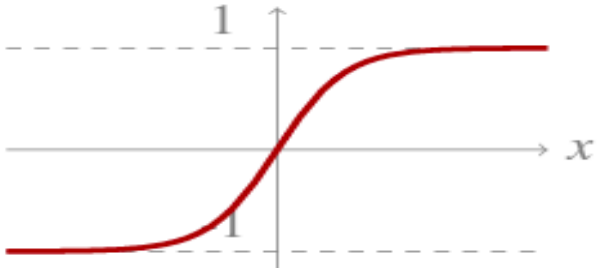
1.1 Redes neurais

As redes neurais artificiais são interconectadas por elementos unitários conhecidos como neurônios, esse nome existe visto que essas redes foram baseadas no funcionamento do cérebro humano. Cada neurônio é constituído pelas entradas, que

são os sinais provenientes de dados que podem ser de dados externos ou dados de demais neurônios, cada neurônio tem também sua função de ativação, a função de ativação recebe a soma de todas as entradas com seus respectivos pesos para cada entrada, esse resultado da entrada passa por uma função de ativação que determina quando a saída do neurônio será ativada ou não, de forma resumida pode-se dizer que a saída do neurônio dependerá se o conjunto de entradas atingirão um valor suficiente para ativar a saída do neurônio ou determinará um valor de saída dependendo da função de ativação do neurônio (BRAGA, LUDEMIR e CARVALHO, 2000).

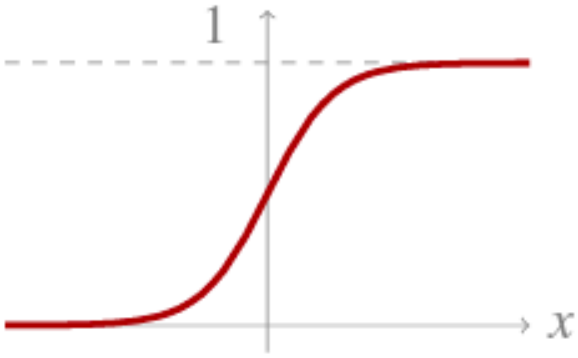
Existem diversos tipos de funções de ativações, pode-se destacar a seguir: **Tanh**: abreviação para tangente hiperbólica e tem sua saída variando conforme a função matemática entre -1 a 1; (PONTI MA, 2017). **Logistic**: representa a função logística com sua variação de 0 a 1; (PONTI MA, 2017). **Relu**: inicia uma rampa ou uma função de segundo grau somente para valores positivos de entradas, caso contrário a função manterá em valor nulo; (PONTI MA, 2017). **Prelu**: é uma função Relu onde é possível ajustar alguns parâmetros tais como a forma de trabalho ocorrem em caso de entradas negativas. (PONTI MA, 2017).

Figura 1 - Representação da tangente hiperbólica



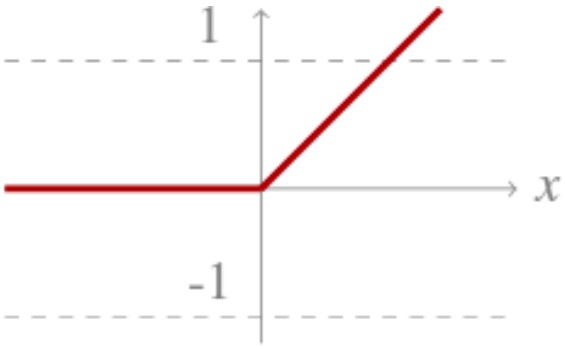
Fonte: (PONTI MA, 2017)

Figura 2 - Representação da função logística



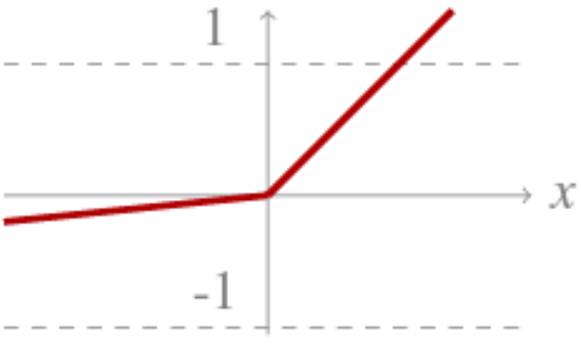
Fonte: (PONTI MA, 2017)

Figura 3 - **Representação da função Relu**



Fonte: (PONTI MA, 2017)

Figura 4 - Representação da função Prelu



Fonte: (PONTI MA, 2017)

Em redes neurais artificiais, os neurônios são dispostos em camadas, cada camada conectada com sua camada antecessora e a camada sucessora, com exceções das camadas de entradas que se conectam aos dados de entrada e a

camada de saída que fornece a saída dos dados. As camadas intermediárias são chamadas de camadas ocultas (PONTI MA, 2017).

1.2 Treinamento de uma rede neural

As formas de se realizar treinamentos em redes neurais artificiais se dividem em duas formas, a de aprendizado supervisionado e de aprendizado não supervisionado (BRAGA, LUDEMIR e CARVALHO, 2000). O aprendizado supervisionado é o mais comum dos treinamentos de redes neurais artificiais, nesse método as entradas e as saídas passam por um supervisor externo, que faz as comparações das entradas e saídas com as saídas esperadas para uma dada entrada, existem alguns algoritmos de correção para reduzir a diferença entre as saídas desejadas e as saídas reais, entre os algoritmos mais utilizados está o *backpropagation*, esse algoritmo classifica o erro da saída da rede e promove os ajustes dos pesos de cada entrada de cada neurônio da rede neural, esse ajuste é baseado na propagação do erro entre as camadas da rede (BRAGA, LUDEMIR e CARVALHO, 2000).

O aprendizado não supervisionado não depende de um supervisor para avaliar as saídas da rede com as saídas esperadas, esse método é utilizado com dados onde não se conhece necessariamente a saída e quando existem várias redundâncias nos dados de entrada de modo que a rede consiga promover classificações dos dados (BRAGA, LUDEMIR e CARVALHO, 2000).

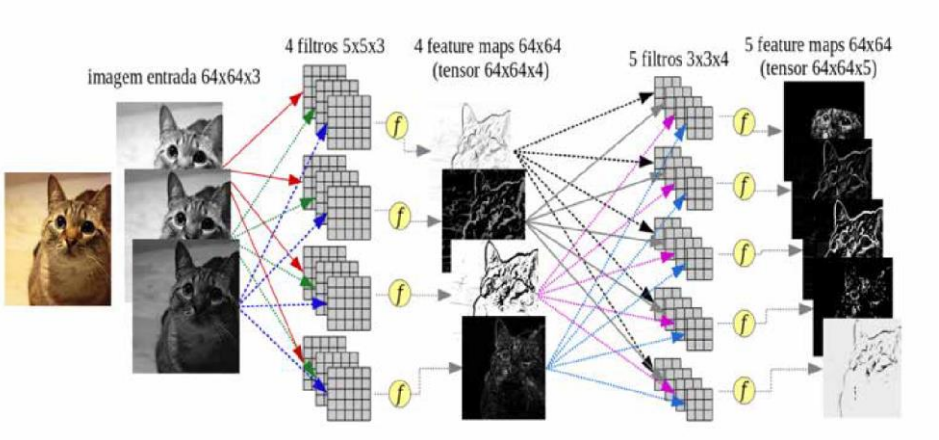
1.3 Redes neurais convolucionais

As redes neurais convolucionais são modelos de redes profundas de aprendizado e são as mais utilizadas recentemente (MAGNANI, FILHO, *et al.*, 2021). Essa rede é caracterizada por ter camadas convolucionais, que camadas ao contrário das camadas de redes densas os neurônios não estão todos conectados com as camadas subsequentes (MAGNANI, FILHO, *et al.*, 2021). O fato de nas redes convolucionais não terem todos os neurônios conectados promove uma grande redução de pesos, isso implica na complexidade de cálculos a serem realizados e possibilita que as redes sejam mais profundas sem perdas de convergência (PONTI MA, 2017) (MAGNANI, FILHO, *et al.*, 2021).

Na camada convolucional, cada neurônio é um filtro aplicado a uma imagem de entrada e cada filtro é uma matriz de pesos. Na Imagem 5, é possível observar a

aplicação de duas camadas de convolução em uma imagem RGB, ou seja, uma imagem com três dimensões. Na primeira camada, 4 filtros 5x5x3 são aplicadas, produzindo 4 mapas de características e, em seguida, outra camada convolucional com 5 filtros 3x3x4, gerar novos mapas de características, esse processo pode ocorrer em várias camadas e em cada camada algumas características específicas são mapeadas (MAGNANI, FILHO, *et al.*, 2021).

Figura 5 - Rede convolucional com duas camadas



Fonte: (PONTI MA, 2017)

1.4 Transfer Learning

Transfer learning é o método onde se aproveita os treinamentos realizados em um dado momento, esse recurso é bastante aplicado pois reduz-se tempo de treinamentos e formação de *dataset*, dessa forma é possível que os pesos das redes já treinadas possam ser reutilizados em uma rede neural e assim as mesmas podem passar por um novo ciclo de treinamento com uma base de dados específicas a fim de conseguir resultados rápidos com pouco treinamento a ser realizado, essa técnica também possibilita a utilização de uma rede neural sem nenhuma necessidade de realização de treinamento, para isso basta que conheça a estrutura da rede utilizada assim como os seus parâmetros (BROWNLEE, 2019).

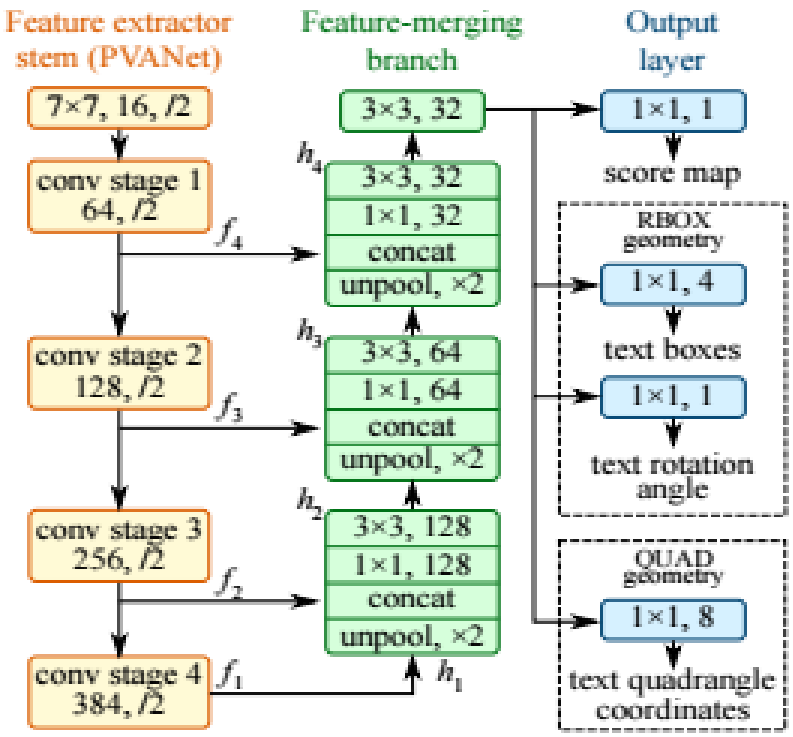
1.5 Redes de identificação de texto

Existem já algumas redes neurais já elaboradas com o foco específico em identificar regiões em imagens que possuem texto, conhecendo essa região pode-se extrair essa imagem e processá-la em um OCR ou em uma rede neural específica para execução de OCR, a rede EAST (*Efficient and Accurate Scene Text Detector*,

Detector de texto preciso e eficiente) que é uma rede que tem a função de localizar em uma imagem onde existe um texto. A rede EAST é uma rede totalmente convolucional proposta em 2017 ela extrai das imagens dimensões (coordenadas) de onde existir textos. A rede EAST é uma rede proposta que se utiliza de camadas convolucionais para encontrar em imagens textos, assim a rede identifica a geometria tais como tamanho, ângulo e coordenadas, a rede também informa um valor de quanto esse segmento da imagem representa um texto ou pode ser considerada um texto. (ZHOU, 2017)

A Imagem 6 representa a estrutura da rede EAST onde é possível ver as pilhas de camadas convolucionais que formam o extrator de características e as que formam as uniões de características.

Figura 6 - Rede EAST



Fonte: (ZHOU, 2017)

2. METODOLOGIA

Este trabalho é uma pesquisa experimental onde se objetiva o estudo e aplicação de uma rede neural capaz de identificar textos em imagens, e terá uma abordagem descritiva pois visa descrever discriminadamente os fatos decorridos no trabalho e qualitativa pois o resultado dos estudos será qualificado quanto sua aplicabilidade e

qualidade final. O objetivo do trabalho é o estudo e aplicação de uma rede neural capaz de identificar textos em imagens.

Para realizar esse desenvolvimento foram utilizados vários recursos, para programar a linguagem escolhida foi a Python, por ser uma linguagem com bastante suporte e por ser mais simples de programar, foram usados também a biblioteca OpenCV que facilita os tratamentos de manipulação de imagens em Python e possibilita carregar modelos de redes neurais assim como os seus pesos, com isso é possível inserir uma imagem na rede e fazer as predições necessárias. O algoritmo desenvolvido foi realizado em um *notebook* Jupyter do Google Colab.

As imagens foram obtidas da internet e de fonte própria de imagens. Essas imagens foram necessárias para poder avaliar e testar os resultados. As imagens abaixo são os códigos utilizados no desenvolvimento do trabalho.

Figura 7 - Rotina principal do programa

```
# define os nomes das duas ultimas camadas da rede para poder retirar os valores finais da camada
layerNames = [
  "feature_fusion/Conv_7/Sigmoid",
  "feature_fusion/concat_3"]

# Fazendo o transfer learning da rede EAST
print("[INFO] Fazendo transfer learning da Rede EAST...")
net = cv2.dnn.readNet('/content/drive/MyDrive/TCC_Engenharia Elétrica/frozen_east_text_detection.pb')

caminho = "/content/drive/MyDrive/TCC_Engenharia Elétrica/imagens_teste"
destino=''
paths = [os.path.join(caminho,fn) for fn in next(os.walk(caminho))[2]]

for img in paths:
  ret=[]
  print("arquivo =",img)
  image = cv2.imread(img)
  ret=detect(img,net)

  nome =os.path.basename(img)
```

Fonte: Autor (2021)

A primeira parte da rotina principal é a definição das camadas da rede neural que serão utilizadas para extrair as informações de predição da rede, essas camadas

fornecerão os dados dos *bounding boxes* e dos scores das imagens. A segunda parte é responsável por fazer a *transfer learning* da rede EAST, o arquivo *frozen_east_text_detection* é o arquivo que contém a estrutura da rede e os seus respectivos pesos, esse arquivo contém os grafos dessa rede, podendo então ser utilizada nos processos de predição adiante. Na parte seguinte é definido a pasta onde estão as imagens que serão testadas na rede neural. Após ler o caminho(*paths*) dos arquivos, os mesmos são arquivados em uma lista que é passada para uma função de detecção descrita a seguir.

Figura 8 - Rotina de detecção de textos

```
def detect(img, net):
  image = cv2.imread(img)
  ret=[]
  orig = image.copy()

  (H, W) = image.shape[:2]

  # define o formato do reshape normalizando a largura e altura
  (newW, newH) = (640, 640)
  rW = W / float(newW)
  rH = H / float(newH)

  # faz o reshape das imagens
  image = cv2.resize(image, (newW, newH))
  (H, W) = image.shape[:2]
  #faz o calculo de medias de cor para melhor definicao de contornos e formas
  avg_color_per_row = np.average(image, axis=0)
  [rr,bb,gg] = np.average(avg_color_per_row, axis=0)

  # extração das camadas rgb ja normalizadas
  blob = cv2.dnn.blobFromImage(image, 1.0, (W, H),
  (rr, bb, gg), swapRB=True, crop=False)
  start = time.time()
  #insere as imagens na camada de entrada da rede neural
  net.setInput(blob)
  #pega o retorno das duas ultimas camadas da rede para pegar a predição da rede
  (scores, geometry) = net.forward(layerNames)
  end = time.time()
  blobb = blob.reshape(blob.shape[2] * blob.shape[1], blob.shape[3], 1)
  cv2_imshow(blobb)

  print("[INFO] A detecção de texto demorou {:.6f} segundos".format(end - start))
  # retira os bb e os seus valores de probabilidade de serem textos
  (numRows, numCols) = scores.shape[2:4]
  rects = []
  confidences = []
  xmax=0.0

  # loop do numeros de linhas
  for y in range(0, numRows):
```

Fonte: Autor (2021)

A função *detect* recebe dois parâmetros de entrada, sendo o primeiro é a lista dos *paths* dos arquivos de imagem e o segundo é a rede neural que foi feito *transfer*

learning. Logo na primeira linha da função a imagem é carregada do arquivo, é então extraído as informações de altura e largura da imagem original, para reduzir e facilitar o processamento foi definido um tamanho padrão de 640 por 640 pixeis, para facilitar e evitar erros de cálculo os tamanhos devem ser múltiplos de 32 já que a rede foi desenhada para trabalhar com tamanhos múltiplos de 32.

Duas variáveis são criadas armazenando os valores de transformação da imagem original com a imagem a ser remodelada em outro formato, essa informação será utilizada mais adiante no código para desenhar os *bounding boxes* na imagem original.

O *resize* faz as imagens serem redimensionada em um outro formato. Uma função da biblioteca OpenCV foi utilizada para efetuarmos a média das cores de cada canal RGB (*Red*, *Blue*, *Green*), essa média é importante pois cada imagem tem uma assinatura em cada canal e é recomendado que faça a normalização dos dados. O Blob é um processamento para extrair as camadas RGB do arquivo e fazer a normalização das imagens conforme as médias de cada canal.

Em *net.setInput* é como se insere a imagem na camada de entrada da rede neural, nesse ponto se inicia o processamento da imagem de entrada e na linha seguinte em *net.foward* faz-se a extração dos dados provenientes do grafos das camadas já definidas na rotina principal, sabe-se que teremos as saídas duas variáveis, uma sendo os scores que são as probabilidades de cada segmento ser um texto e os dados dos segmentos dos textos.

Em seguida alguns processamentos de dados são realizados com o fim de realizar a leitura dos dados das geometrias dos segmentos de imagens, os dados retirados de números de linhas e número de colunas serão necessários para que adiante possa-se pegar cada segmento de imagem e medir seu respectivo score.

Duas listas foram criadas para armazenar os retângulos e as probabilidades ou *scores* de cada retângulo, lembrando que uma imagem pode possuir várias áreas que tem textos presentes. Por fim inicia-se uma rotina na linha 40 que vai correr todo o vetor de linhas que será descrito adiante.

Figura 9 - Loop nos números de linhas, colunas e tratamento das imagens

```
    #leitura das probabilidades e formas geometricas, usados para bb dos textos
    scoresData=scores[0,0,y]
    xData0 = geometry[0,0,y]
    xData1 = geometry[0,1,y]
    xData2 = geometry[0,2,y]
    xData3 = geometry[0,3,y]
    anglesData = geometry[0,4,y]
    for x in range(0,numCols):
     if(xmax<scoresData[x]):
       xmax=scoresData[x]

# se o score de probabilidade de texto for menor que 50% será descartado
     if (scoresData[x] <0.5):

       continue

     #calculo de offset como resultado do mapa de caracteristicas
     (offsetX,offsetY) = (x*4.0,y*4.0)
     #conversao dos angulos dados do bb
     angle = anglesData[x]
     #print("angulo =",angle)
     cos = np.cos(angle)
     sin = np.sin(angle)
     #calculo de altura e largura do bb
     h = xData0[x]+xData2[x]
     w = xData1[x]+xData3[x]
     #print("h =",angle)
     #print("w =",angle)
     #calculo dos inicios e finais do bb
     endX = int(offsetX + (cos * xData1[x])+(sin * xData2[x])+10)
     endY = int(offsetY - (sin * xData1[x])+(cos * xData2[x])+10)
     startX=int(endX -w-10)
     startY=int(endY -h-10)

     rects.append((startX,startY,endX,endY))
     confidences.append(scoresData[x])
```

Fonte: Autor (2021)

Para o looping do número de linhas retira-se os dados de coordenadas de cada retângulo dos segmentos de imagem assim como o seu ângulo. No *loop* de colunas vamos ter cada *score* de cada retângulo, a seguir verificamos se o *score* do retângulo é maior que 0,5 isso significa que somente irá aproveitar os retângulos que tem mais de 50% de chances de serem segmentos de imagens com texto. A seguir os processamentos matemáticos para que seja possível converter os retângulos das imagens comprimidas com seu formato ajustado aos ângulos e direção dos textos,

visto que os textos podem estar inclinados ou em uma direção não convencional assim como é considerado o efeito de profundidade que algumas imagens possuem.

Figura 10 - Loop nos números de linhas, colunas e tratamento das imagens

```
79   print("xmax=",xmax)
80   confidences, rects = zip(*sorted(zip(confidences, rects)))
81
82 # Faz o nommaxsuppression para formar bb que se sobrepoem, e aproveita os 5 maiores
83
84   boxes = non_max_suppression(np.array(rects), probs=confidences)[0:7]
85   cv2_imshow(orig)
86   print("[INFO] foram identificados {:.6f} imagens".format(len(boxes)))
87
88 # loop bb
89   te=0;
90   for (startX, startY, endX, endY) in boxes:
91     if(startX<1):
92       continue
93     if(startY<1):
94       continue
95
96     te=te+1
97   # faz o rescaling dos bb para ajustar na imagem original
98
99     startX = int(startX * rW)
100    startY = int(startY * rH)
101    endX = int(endX * rW)
102    endY = int(endY * rH)
103
104  # sobrepoe o bb na imagem original
105    cv2.rectangle(orig, (startX, startY), (endX, endY), (0, 255, 0), 2)
106    crop=orig[startY:endY,startX:endX]
107    ret.append(crop)
108    #print(crop.shape)
109  cv2_imshow(orig)
110
111  return ret
```

Fonte: Autor (2021)

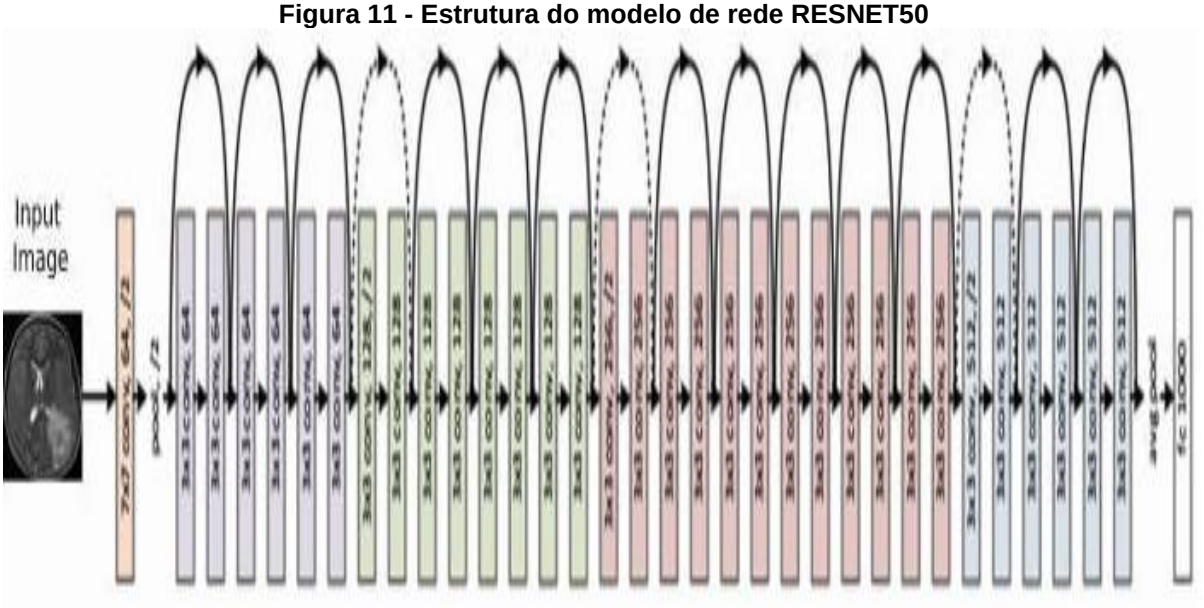
A linha 80 faz uma ordenação das duas listas, sendo a lista de probabilidades e a lista dos retângulos, essa ordenação não é importante, ela foi feita para poder facilitar o tratamento do *non_max_supressor*, essa função converte vários retângulos que se sobrepõe em um retângulo apenas, isso é recomendado para evitar que um segmento

de texto seja considerado vários segmentos de talvez contendo um único caractere. Junto com o *non_max_supressor* faz-se outro corte do vetor pegando somente os 7 maiores segmentos de texto considerando a sua probabilidade, isso foi feito para reduzir a quantidade de retângulos em imagens com muitos textos.

Após esse processamento o *rescaling* é feito para poder ajustar os retângulos nas imagens de tamanho original, lembrando que as imagens foram processadas para reduzir o custo computacional.

Explicado o algoritmo é necessário explicar a rede neural utilizada, nesse caso foi utilizada uma rede neural EAST cujo sua customização foi realizada utilizando como base a RESNET50, ou seja, quando a rede foi desenvolvida e ajustada para funcionar no método EAST o autor original se fez valer também de *transfer learning* de uma rede convolucional já amplamente estudada e utilizada. Abaixo é exibido a estrutura básica da rede Resnet 50 que é a base convolucional da rede EAST.

Figura 11 - Estrutura do modelo de rede RESNET50



Fonte: (HASSAN ALI KHAN, 2020)

O modelo da rede EAST pode ser obtido da internet para realizar o processo de *transfer learning*, dessa forma o modelo foi obtido.

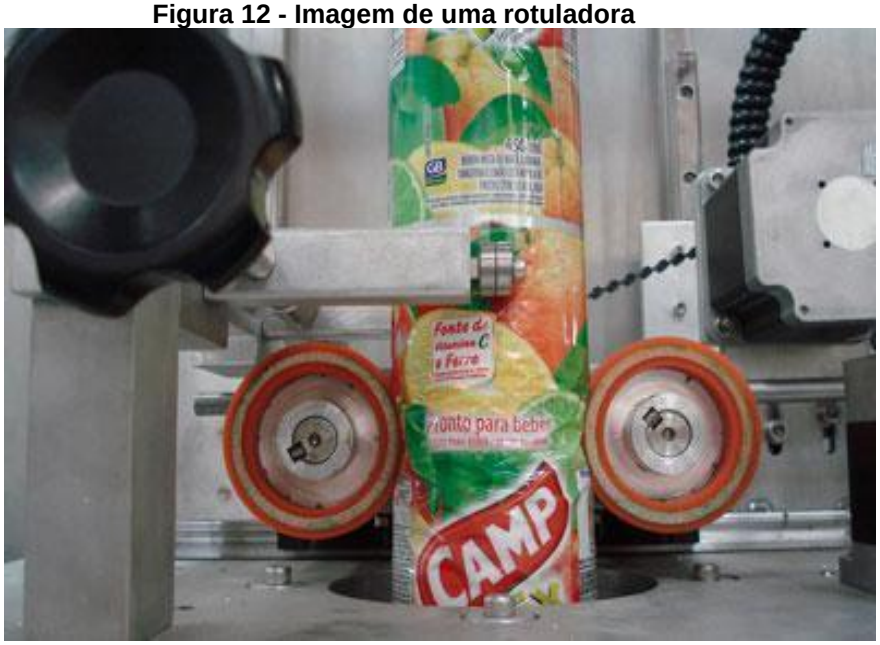
3. RESULTADOS E DISCUSSÕES

Com o programa pronto e os métodos e ferramentas já elaboradas, a etapa seguinte é a aplicação do trabalho com imagens reais e verificar os resultados, nessa etapa algumas imagens foram selecionadas para realizar o teste. As imagens selecionadas estão a seguir.

Figura 11 - Foto de uma rua

Fonte: (GOOGLE, 2020)

Figura 12 - Imagem de uma rotuladora



Fonte: (MKM, 2021)

Figura 13 - Imagem de uma linha de produção

Fonte: (REFRIGERANTE CONVENÇÃO, 2021)

Figura 14 - Imagem de uma esteira com pacotes

Fonte: (EPOCANEGOCIOS, 2019)

Essas imagens serão utilizadas e foram adquiridas aleatoriamente para realizar os testes. Como parte analítica do trabalho escolheu uma imagem para verificar o blob dos canais de cores RGB.

Figura 15 - Blob de uma imagem de uma rua

Fonte: Autor (2021)

Essa é a representação da imagem que será a entrada da rede neural, a seguir temos a saída como resultado do processamento da imagem, nesse caso foram encontradas 3 possíveis regiões de texto.

Figura 16 - Resultado de identificação em uma imagem de uma rua

Fonte: Autor (2021)

Como pode-se observar a imagem tem 3 possíveis textos, sendo que dois deles são facilmente reconhecidos das marcas Itaú e Bradesco, um segmento da parede do prédio foi erroneamente identificado como um texto. O resultado da análise da segunda imagem está a seguir.

Figura 17 - Resultado de identificação em uma imagem de rotuladora

Fonte: Autor (2021)

Como pode-se observar a imagem tem 4 possíveis textos, todos identificados corretamente, pode-se observar que alguns textos não foram identificados corretamente, esse ponto se dá ao fato de alguns textos terem probabilidades baixas na predição. O resultado da análise da terceira imagem está a seguir.

Figura 18 - Imagem refrigerantes em uma linha de produção

Fonte: Autor (2021)

Como pode-se observar a imagem tem 7 possíveis textos, todos identificados corretamente, pode-se observar que alguns textos não foram identificados corretamente, lembrando que para poder avaliar a performance foram limitados a 7 possíveis textos. O resultado da análise da quarta imagem está a seguir.

Figura 19 - Imagem de uma esteira com pacotes

Fonte: Autor (2021)

Como pode-se observar a imagem tem 7 possíveis textos, todos identificados corretamente, pode-se observar que alguns textos não foram identificados corretamente, lembrando que para poder avaliar a performance foram limitados a 7 possíveis textos.

4. CONSIDERAÇÕES FINAIS

Por fim pode-se constatar que usando técnicas modernas de inteligência artificial e usando de recursos de otimização como *transfer learning* pode otimizar tempo de desenvolvimento e treinamento. Sendo possível e prático desenvolver sistemas de identificação de texto em imagens nas mais diversas situações. Os resultados foram satisfatórios e existem pontos de melhorias identificados, recomenda-se um estudo mais aprofundado para avaliar os motivos por existirem segmentos de imagens que foram falsos positivos, deve-se avaliar também futuramente pontos de melhorias desse sistema para detecção de alguns textos que não foram bem identificados.

O sistema desenvolvido foi testado em outras imagens com resultados semelhantes, porém devido ao volume de dados foram escolhidas essas 4 imagens

para avaliar o desempenho e discussões, esse sistema também pode ser utilizado em sistemas de monitoramento em tempo real em diversas aplicações tais como rotuladoras e seleção de pacotes em esteiras transportadoras.

5. REFERÊNCIAS

BRAGA, A. P.; LUDEMIR, T. B.; CARVALHO, A. C. P. L. F. **Redes Neurais Artificiais:** Teoria e Aplicações. 1. ed. Rio de Janeiro: LTC, 2000.

BROWNLEE, J. A Gentle Introduction to Transfer Learning for Deep Learning. **machinelearningmastery.**, 2019. Disponível em: <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>. Acesso em: 1 out. 2021.

EPOCANEGOCIOS. Amazon lança o Amazon Prime no Brasil. **Época Negócios**, 2019. Disponível em: <https://epocanegocios.globo.com/Empresa/noticia/2019/09/amazon-lanca-o-amazon-prime-no-brasil.html>. Acesso em: 1 out. 2021.

GOOGLE. Google Street. **Google Street**, 2020. Disponivel em: <www.google.com.br>. Acesso em: 15 set. 2021.

HASSAN ALI KHAN, W. J. M. M. M. U. M. Brain tumor classification in MRI image using convolutional neural network. **Mathematical Biosciences and Engineering**, v. 17, p. 6203-6216., 2020. ISSN doi: 10.3934/mbe.2020328.

MAGNANI, M. J. H. et al. APPLICATION OF DEEP LEARNING FOR ANALYSIS OF CRACKS IN PELLET FALLING. In: OLIVEIRA, A. C. D. **Coleção desafios das engenharias:** engenharia de computação. 1. ed. Ponta Grossa: Atena, 2021. Cap. 10, p. 388.

MENDES, M. HISTÓRIA DA ESCRITA. **Educa+Brasil**, 2020. Disponível em: <https://www.educamaisbrasil.com.br/enem/lingua-portuguesa/historia-da-escrita>. Acesso em: 01 out. 2021.

MKM. Máquina Rotuladora. **mkmsp**, 2021. Disponível em: <https://www.mkmsp.com.br/maquina-rotuladora.php>. Acesso em: 2 out. 2021.

PONTI MA, C. G. D. Como funciona o deep learning. In: SBC **Tópicos em gerenciamento de dados e informações**. Uberlandia: [s.n.], 2017. Disponível em: <http://sbbd.org.br/2017/wpcontent/uploads/sites/3/2017/10/topicos-em-gerenciamento-de-dados-e-informacoes-2017.pdf>.

REFRIGERANTE CONVENÇÃO. Refrigerante Convenção. **Refrigerante Convenção**, 2021. Disponível em: <https://www.refrigerantesconvencao.com.br/>. Acesso em: 2 out. 2021.

ROSEBROCK, A. OpenCV Text Detection (EAST text detector). **pyimagesearch**, 2018. Disponível em: <https://www.pyimagesearch.com/2018/08/20/opencv-text-detection-east-text-detector/>. Acesso em: 15 set. 2021.

SCHULTZ, F. Conheça as 7 vantagens da automatização de processos. **Bom Controle**, 2020. Disponível em: <https://blog.bomcontrole.com.br/vantagens-automatizacao-de-processos/>. Acesso em: 2 out. 2021.

ZHOU, X. *et al*., "EAST: An Efficient and Accurate Scene Text Detector," in 2017 IEEE **Conference on Computer Vision and Pattern Recognition** (CVPR), Honolulu, HI, USA, 2017 pp. 2642-2651. Acesso em: 2 out. 2021.