UNIVERSIDADE LUTERANA DO BRASIL

PRÓ-REITORIA DE GRADUAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Felipe Szczesny Sperb

Visão Computacional Aplicada na Detecção de Equipamentos de Proteção Individual

Canoas, Novembro de 2021

Visão computacional aplicada na detecção de equipamentos de proteção individual

Felipe Szczesny Sperb
Departamento de Engenharia Elétrica
Universidade Luterana do Brasil (ULBRA)
ULBRA
Canoas/RS, Brasil
felipe.sperb@rede.ulbra.br

***Resumo* - Este artigo se refere a pesquisa e ao desenvolvimento de um protótipo capaz de detectar equipamentos de proteção individual (EPI) em trabalhadores da indústria e da construção civil, utilizando técnicas de visão computacional e Redes Neurais Convolucionais (Convolutional Neural Network – CNN). A arquitetura escolhida para a CNN foi a YOLOv4 (You Only Look Once) com o framework Darknet. A aplicação foi desenvolvida para ser processada em CPU (Central Processing Unit), utilizando linguagem de programação Python e com o auxílio de bibliotecas como OpenCV e MediaPipe. O protótipo foi capaz de detectar EPIs, quando corretamente posicionados no corpo do trabalhador, a uma distância de até três metros, com uma precisão de 96,69% e com uma cobertura de 82,16%.**

Palavras-chave* - *equipamentos de proteção individual, detecção de objetos, redes neurais convolucionais, YOLO.

I. INTRODUÇÃO

Segundo relatório do Ministério Público do Trabalho (MPT) em conjunto da Organização Internacional do Trabalho (OIT), publicado no ano de 2021, o Brasil registrou entre os anos de 2002 e 2020 uma taxa de 6 óbitos a cada 100 mil empregos formais, colocando o país na segunda posição em mortalidade no trabalho entre os países do G20. Em oito anos foram registrados no Brasil 5,6 milhões de doenças e acidentes de trabalho, que geraram um gasto previdenciário que ultrapassa R$100 bilhões. De 2012 a 2020, a maior parte dos acidentes no país, 15%, foram ocasionados pela operação de máquinas e equipamentos. No mesmo período, entre as lesões mais frequentes, 900 mil casos são lacerações, feridas contusas (produzida por golpe ou impacto) e puncturas (furos e picadas). Na sequência aparecem as fraturas (mais de 750 mil), contusões e esmagamentos (mais de 659 mil), distensões e torções (mais de 390 mil) e lesões imediatas (mais de 380 mil). No mundo, estima-se que doenças e acidentes do trabalho produzam a perda de 4% do PIB global a cada ano. Em países em desenvolvimento esse número pode chegar a 10% [1] [2].

Acidentes de trabalho podem ser de dois tipos: acidentes atípicos, que correspondem a acidentes no percurso do trabalho, por exemplo, ou acidentes típicos, ou seja, que estão diretamente ligados com a função do trabalhador. Pelo menos três quartos dos acidentes registrados no Brasil estão no último grupo [3]. Uma parte desses acidentes poderiam ser evitados pelo uso correto de Equipamentos de Proteção Individual (EPI). Os EPIs são equipamentos que ajudam a evitar ou amenizar o risco de acidentes de trabalho. Eles também podem prevenir que o profissional adquira alguma doença em decorrência da função laboral ou contamine um produto em uma linha de produção, por exemplo. No Brasil, a norma técnica que regulamenta o uso de EPIs é a NR6 [4].

Mas mesmo em empresas que realizam treinamento e disponibilizam EPIs aos seus funcionários, ainda são registrados acidentes. Entre os principais fatores do uso incorreto dos Equipamentos de Proteção Individual estão a negligência e a imprudência. O excesso de confiança na execução de uma tarefa pode fazer com que o profissional ignore o procedimento padrão, por exemplo. Mas em outros casos, pode existir até mesmo uma limitação física inerente ao ser humano envolvida. Conceitos muito conhecidos na neurociência e na psicologia cognitiva como a Atenção Seletiva [5] e a Cegueira à Mudança [6], demonstram que quando centramos a atenção em uma tarefa tendemos a nos tornar "cegos" a alterações do ambiente e a perder informações relevantes. Esses mesmos fenômenos, muito explorados por ilusionistas em suas performances, podem fazer com que um funcionário de manutenção, sob pressão para recolocar em funcionamento uma máquina que parou de repente a linha de produção, não perceba que seus óculos de proteção estão pendurados no pescoço enquanto ele executa a tarefa, por exemplo. Mesmo profissionais que passaram a vida inteira tomando todos os cuidados e realizando todos os procedimentos corretamente estão suscetíveis a cometer um erro, que infelizmente pode ser fatal.

A proposta deste trabalho visa desenvolver um sistema que possa auxiliar nesses processos de segurança e que possa detectar a ausência ou a presença dos EPIs que o trabalhador deveria estar vestindo. Para isso foram exploradas técnicas de visão computacional e redes neurais convolucionais (Convolutional Neural Network – CNN) para detecção de objetos. Buscando auxiliar empresas e funcionários a respeitarem as normas de segurança e zelar pela saúde do profissional.

II. ESTRUTURA DE UM SISTEMA DE VISÃO COMPUTACIONAL

O método de processamento de imagens digitais decorre de duas áreas principais: melhoria de informação visual para a interpretação humana e o processamento de dados de cenas para percepção automática através de máquinas [7]. Frequentemente os autores denominam a primeira área como 'processamento de imagens' e a segunda área como 'visão computacional', 'visão artificial', 'reconhecimento de padrões' ou 'análise de imagens'.

Podemos definir um sistema de visão computacional como um sistema capaz de adquirir, processar e interpretar imagens correspondentes a cenas reais [8]. A Fig. 1 demonstra as principais etapas de um sistema tradicional de análise de imagens. Nas seções seguintes será exemplificado como cada bloco está sendo abordado neste trabalho.

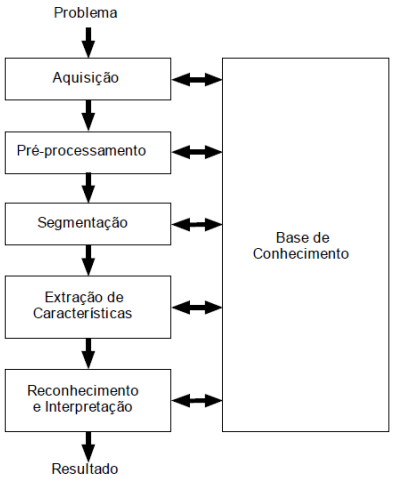

Fig. 1. Sistema de visão computacional e suas principais etapas. Imagem disponivel em [8].

III. Problema

Neste projeto, o problema consiste em identificar os EPIs que estão sendo vestidos por uma pessoa em uma cena. Na Fig. 2 estão ilustrados os sete equipamentos de segurança que foram arbitrariamente definidos: máscara, capacete, óculos, protetor auditivo circum-auricular, luva, bota e colete refletivo.

Fig. 2 Exemplo de equipamentos de proteção indivisual utilizados no estudo. a) máscara PFF2 N95; b) capacete sem jugular; c) óculos de proteção; d) abafador acoplável; e) luva de poliamida com banho nítrico; f) botina e g) colete de proteção refletivo. Imagens disponíveis em [9].

IV. Aquisição de imagem

A aquisição da imagem será feita por uma câmera webcam HD $1280p \; x \; 720p$, padrão de um notebook Acer® Aspire E1-531, capturando imagens em vídeo, frame a frame, a uma taxa média de $10 \; FPS$. Os testes iniciais foram realizados com a câmera em duas posições: à $90cm$ da pessoa (Fig. 3), capturando imagem do tórax até a cabeça, e à $3m$ da pessoa (Fig. 4), capturando imagens dos pés à cabeça.

Fig. 3 Cena com câmera posicionada à 90cm de distância. Imagem gerada pelo autor.

Fig. 4 Cena com câmera posicionada à 3m de distância. Imagem gerado pelo autor

O sistema deve ser capaz de analisar diferentes tipos de cenas. Algumas mudanças de luminosidade, ruido, intensidade ou número de pixels podem afetar a analise, porém se espera que algum nível de mudança seja completamente tolerado. Pode-se observar a diferença dos histogramas da Fig. 3 e da Fig. 4 nas Fig. 5 e Fig. 6, respectivamente. Ambas imagens com objetos detectados.

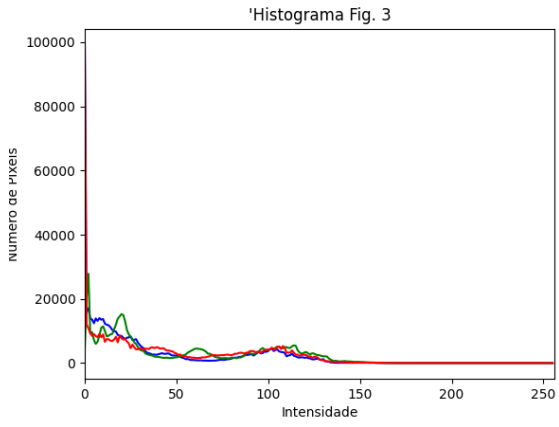

Fig. 5 Histograma RGB da Fig. 3, cena em ambiente interno com iluminação baixa. Gráfico gerado pelo autor.

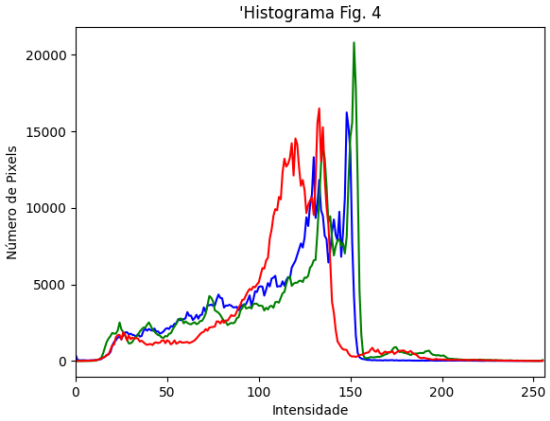

Fig. 6 Histograma RGB da Fig. 4, cena em ambiente externo com luz natural. Gráfico gerado pelo autor.

V. PRÉ-PROCESSAMENTO E SEGMENTAÇÃO

A função da etapa de pré-processamento é aprimorar a qualidade da imagem para as etapas subsequentes enquanto que a tarefa básica da segmentação é a de dividir uma imagem em suas unidades significativas [8]. Nesse projeto as etapas de pré-processamento e de segmentação estão entrelaçadas. Algumas técnicas não exploradas como o ajuste automático de contraste e brilho, por exemplo, podem ser importantes para melhoria do protótipo no futuro e serão indicadas no item XI. Somente serão enviadas para a CNN as imagens com a presença de uma pessoa, isso será feito utilizando uma técnica de estimativa de postura. Imediatamente antes da imagem entrar na CNN ela é transformada em um objeto BLOB (Basic Large Object). A estimativa de postura é novamente utilizada na saída da CNN com o objetivo de comparar o resultado obtido com a região de interesse (Region of Interest – ROI)

A. *Objeto BLOB*

Um BLOB é um objeto numpy 4D (imagens, canais, largura e altura). O objetivo desta etapa é de que todas as imagens, independentemente da câmera que está sendo utilizada, ao entrar na rede: tenham as mesmas dimensões espaciais ($416x416$), tenham a mesma profundidade (número de canais) e possuam o mesmo valor de escala para redimensionar cada pixel na faixa de 0 a 1 ($1/255$) [10]. Na Fig. 7 está representada a reconstrução de um canal da Fig. 4 em formato de objeto BLOB.

Fig. 7 Reconstrução da imagem da Fig. 4 em imagem BLOB. Imagem gerada pelo autor.

B. *Estimativa de postura humana*

A estimativa de postura humana é uma técnica de Machine Learning (ML) que busca a localização das articulações humanas (cotovelos, pulsos, etc.) ou uma posição especifica do corpo em uma imagem ou vídeo. Nesse trabalho foi utilizada uma biblioteca de código aberto do Google® chamada MideaPipe Pose. Está é uma solução de alta fidelidade para rastreamento de postura do corpo humano e com bom desempenho em CPU (Central Processing Unit) a taxas médias que variam de 10 FPS à 30 FPS (dependendo do hardware). A biblioteca foi treinada para fornecer 33 landmarks, pontos de referência 3D (Fig. 8), e máscara de segmentação de fundo em todo o corpo a partir de quadros de vídeo RGB (Fig. 10). Cada ponto localizado retorna as coordenadas x, y e z, onde a coordenada z é referenciada pela posição do ponto em relação ao tronco do corpo, sendo uma variável importante para definir a visibilidade do landmark [11].

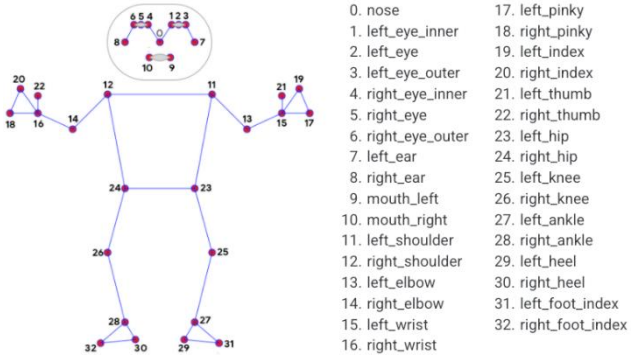

Fig. 8 Pontos de referencia para biblioteca MideaPipe (pose landmarks). Imagem disponível em [11].

Fig. 9 Estimativa de postura da Fig. 4. Imagem gerada pelo autor.

Fig. 10 Máscara de segmentação de fundo aplicada na Fig. 4. Imagem gerada pelo autor.

O pipeline da solução funciona em duas etapas: primeiro é utilizado um detector de face para localização da ROI dentro da imagem, posteriormente um algoritmo de rastreamento prevê os landmarks e a máscara de segmentação dentro da ROI, usando o quadro recortado da ROI como entrada. O detector é invocado apenas quando necessário, no primeiro frame ou quando o rastreador não consegue identificar a presença de pose no frame anterior. Para os outros frames, o pipeline simplesmente deriva a ROI dos landmarks de pose do frame anterior (Fig. 11) [11] [12].

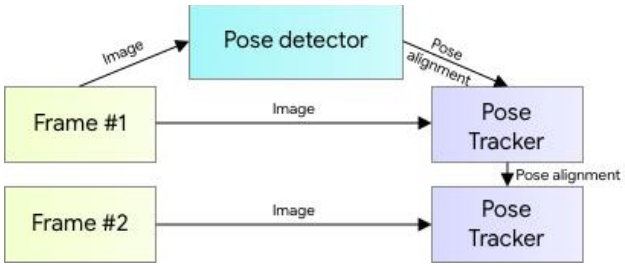

Fig. 11 Pipeline de estimativa de postura. Imagem disponível em [12].

Com essa técnica foi possível arbirtrar uma posição de inspeção, assim a imagem só seria enviada para etapa de detecção de objetos (muito mais custosa em recursos computacionais) se a postura correta for detectada. Para isso foi avaliado o ângulo formado pelos landmarks correspondentes ao ombro, ao cotovelo e ao pulso de cada braço. A posição é dada como correta quando o resultado de (1) ficar entre -160° e $-20°$ para o braço direito e entre 160° e 20° para o braço esquerdo. Importante salientar que essa técnica para obter uma posição de inspeção pode não ser eficiente para deficientes fisicos de membros superiores. Desta forma, para a implementação desse sistema, deve-se encontrar alternativas que incluam esses casos, como uma outra posição auxiliar ou um comando por voz, por exemplo.

$$\theta = \tan^{-1}\left(\frac{y_{pulso}-y_{cotovelo}}{x_{pulso}-x_{cotovelo}}\right) - \tan^{-1}\left(\frac{y_{ombro}-y_{cotovelo}}{x_{ombro}-x_{cotovelo}}\right) \quad (1)$$

Comparando os landmarks da imagems com as coordenas do objeto detectado é possível avaliar se a posição do objeto está correta. Na Fig. 12, por exemplo, observa-se que a máscara está cobrindo os landmarks referentes ao nariz e aos cantos da boca (de acordo com a Fig. 8, os pontos 0, 9 e 10), portanto caso a máscara seja detectada em uma posição que não conhecida com estes pontos, o sistema pode indicar que o EPI está mal posicionado. Observa-se que como a pipele da MidePipe inicia com a detecção de face e como a máscara cobre parcialmente o rosto, podem haver pequenos desvios na detecção dos landmarks, mas nada tão critico. Já na Fig. 13 a máscara está mal posicionada, não coincidindo com a ROI que esse EPI deve ocupar. Nesse caso o sistema deverá detectar essa discrepância.

A técnica ainda pode ser falha caso a pessoa simplesmente segure a máscara na frente do rosto a uma certa distância (Fig. 14), assim o sistema está suscetivel a não detectar que o EPI está mal posicionado. Mas ainda assim, acredita-se que essa técnica é capaz de diminuir consideravelmente os falsos positivos e melhorar o desempenho do sistema.

Fig. 12 Exemplo em que a posição da máscara coincide com os landmarks referentes ao nariz e aos cantos da boca. Imagem gerada pelo autor.

Fig. 13 Exemplo em que a posição da máscara não coincide com os landmarks referentes ao nariz e aos cantos da boca. Imagem gerada pelo autor.

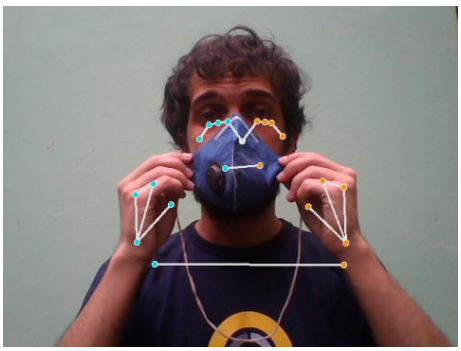

Fig. 14 Exemplo em que o sistema pode ser "enganado", pois apesar da máscara estar posicionada nas mesmas coordenadas dos landmarks referentes ao nariz e aos cantos da boca, o EPI está apenas sendo segurado na frente do rosto, não estando adequadamente vestido. Imagem gerada pelo autor.

VI. Extração de caracteristicas

Nessa etapa, cada imagem analisada passará por uma Rede Neural Convolucional. As CNNs são redes neurais que usam a operação matemática de convolução no lugar da multiplicação geral de matrizes em pelo menos uma de suas camadas [13]. Inicialmente o modelo foi proposto no ano de 1998 por Yann Lecun na detecção de caracteres [14]. A partir do ano de 2012 a técnica passou a ser largamente utilizada em projetos de detecção de objetos [15].

A convolução é uma operação que realiza o somatório do produto entre duas funções distintas, ao longo da região em que elas se sobrepõem, em razão do deslocamento existente entre elas. No contexto da visão computacional, uma destas funções será uma imagem representada como uma matriz multidimensional de dados, enquanto a outra função será um filtro de extração de características, que chamaremos de kernel, representado por uma matriz

multidimensional de parâmetros que serão adaptados pelo algoritmo de aprendizagem. Durante a aplicação da convolução o kernel se desloca ao longo da imagem segundo um número determinado de passos (ou strides) [16] [13].

Considerando uma imagem I e um kernel K com dimensões $m\ x\ n$, a operação pode ser definida em (2).

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \quad (2)$$

A convolução potencializa três ideias importantes que podem ajudar a melhorar um sistema de ML: interações esparsas, compartilhamento de parâmetros e representações variáveis. Além disso, a convolução fornece um meio de trabalhar com entradas de tamanho variável. Essas características reduzem os requisitos de memória e melhoram a eficiência estatística da rede [13].

As CNNs especializadas na detecção de objetos possuem diversas arquiteturas com diferentes características e desempenho. Para realização deste trabalho foi escolhido o detector de um estágio YOLOv4 (You Only Look Once) devido seu equilíbrio entre velocidade e precisão e o fato de que esta é uma arquitetura de código aberto.

A primeira versão do YOLO foi proposta em 2016 por Joseph Redmon [17]. Redmon desenvolveu a arquitetura até sua terceira versão [18] [19]. A quarta versão, utilizada neste projeto, foi proposta em 2020 por Alexey Bochkovskiy [20]. Apesar de não apresentar grandes melhorias em velocidade, a arquitetura apresenta melhorias significativas em precisão.

A arquitetura do YOLOv4, conforme Fig. 15, é dividida em cinco partes:

A. Input:

É simplesmente a imagem de entrada da rede. Todas as imagens são configuradas para $416x416$ pixels.

B. Backbone:

O backbone é baseado em CSPDarknet53, um framework de código aberto para redes neurais. É nesta etapa em que a rede é treinada. Seu objetivo principal é o de extrair características essenciais da entrada.

O backbone é dividido em três blocos, onde cada um irá gerar uma detecção na última etapa. O primeiro bloco possui 94 camadas convolucionais e os dois últimos possuem mais 8 camadas cada. Em cada camada é possível configurar alguns parâmetros:

- *Batch normalize:* é possível definir se será ou não utilizada a normalização de lote. Para aumentar a estabilidade de uma rede neural, a normalização dos lotes normaliza a saída de uma camada de ativação anterior, subtraindo a média do lote e dividindo pelo desvio padrão do lote [21]. Por padrão, a única camada que não está utilizando é a última camada de cada bloco.
- *Filters:* é o número de kernels. Esse número definirá a quantidade de feature maps na saída de cada camada convolucional. A maioria das camadas possuem 32, 64, 128, 256, 512 ou 1024 kernels. Na última camada de cada bloco, o número de kernels foi ajustado para 36 conforme (3), onde $classes$ se refere ao número de objetos treinados, ou seja, sete.

$$filters = 3(classes + 5) \quad (3)$$

- *Size:* é o tamanho do kernel. Nas camadas convolucionais se encontra kernels de dimensões $1x1$ e $3x3$.
- *Stride:* é o número de passos. Por padrão, o YOLOv4 utiliza um ou dois passos.
- *Pad:* define se será usado padding na camada. Ou seja, se haverá compensação de pixels nas bordas da imagem após a convolução. Em cada camada o padding é definido conforme (4).

$$padding = \frac{size}{2} \quad (4)$$

- *Activation:* define a função de ativação. Esse mecanismo permite que pequenas mudanças nos pesos e no bias da rede neural causem apenas uma pequena alteração na saída [22]. Uma característica do YOLOv4 é o uso da função mish na maioria das camadas, com uma pequena quantidade de informação negativa.

A função de perda utilizada nessa rede neural é $CIoU$ (Complete Intersection over a Union) [23], representada em (5).

$$L_{CIoU} = 1 - IoU + \frac{p^2(A,B)}{c^2} + \alpha v \quad (5)$$

Onde IoU é a intersecção sobre a união de duas caixas (A e B), onde uma representando a posição verdadeira do

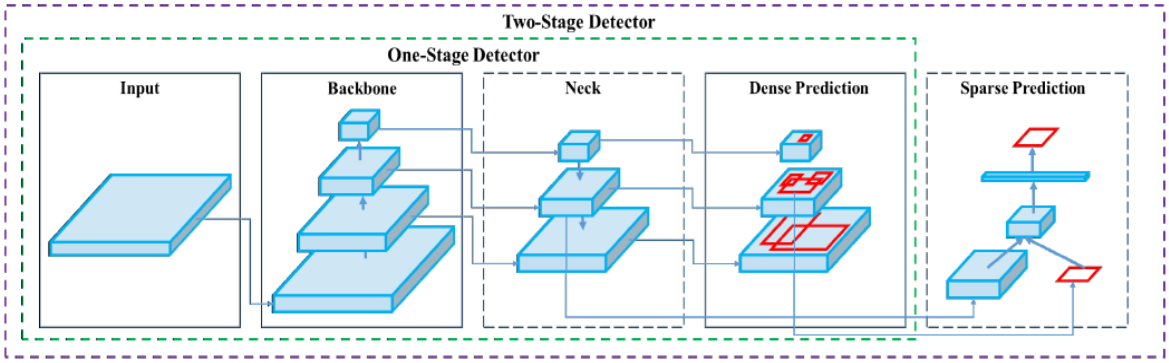

Fig. 15 Arquitetura YOLOv4. Imagem disponível em [20].

objeto e a outra representa a previsão realizada (6), respectivamente.

$$IoU = \frac{A \cap B}{A \cup B} \quad (6)$$

Ainda em (5), $p^2(A, B)$ é a distância euclidiana entre os pontos centrais das duas caixas, c é o comprimento diagonal da menor caixa que cobre duas caixas, v mede a consistência da proporção e α é um parâmetro de compensação [23].

$$v = \frac{4}{\pi^2}(\tan^{-1}\frac{w_B}{h_b} - \tan^{-1}\frac{w_A}{h_A})^2 \quad (7)$$

$$\alpha = \frac{v}{(1-IoU)+v} \quad (8)$$

Onde em (7) w e h representam respectivamente a largura e a altura de cada caixa.

C. *Neck:*

O objetivo desta etapa é compor informações das diferentes camadas do backbone para os detectores. As técnicas que a arquitetura utiliza são SPP (Spatial Pyramid Pooling Layer) e PaNet [20].

D. *Head:*

As duas últimas etapas são compostas pelo detector de um estágio já utilizado no YOLOv3. São previstas detecções em três escalas diferentes. A imagem é dividida em 64 células, 256 células e 1024 células. Cada célula possui dimensões $cx\ x\ cy$ e prevê B caixas delimitadoras e pontuações de confiança para essas caixas. Essas pontuações de confiança refletem a confiança do modelo de que a caixa contém um objeto e também o quão precisa ela pensa que a caixa é prevista. Formalmente, definimos a confiança em (9). Se nenhum objeto existir nessa célula, as pontuações de confiança devem ser zero. Caso contrário, queremos que a pontuação de confiança seja igual à (6).

$$\Pr(Objeto) * IoU = \sigma(t_o) \quad (9)$$

É realizada uma previsão densa, composta por um vetor contendo o rótulo do objeto, a pontuação de confiança e as coordenadas da caixa delimitadora prevista, descritas em (10), (11), (12) e (13) e detalhadas da Fig. 16.

$$b_x = \sigma(t_x) + c_x \quad (10)$$

$$by = \sigma(t_y) + c_y \quad (11)$$

$$b_w = p_w e^{t_w} \quad (12)$$

$$b_h = p_h e^{t_h} \quad (13)$$

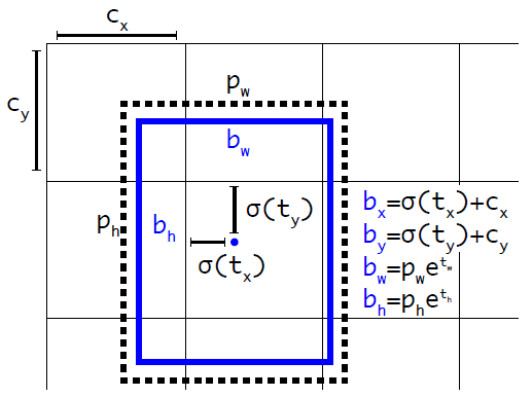

Fig. 16 Detalhamento das variáveis da caixa de detecção de objeto. Imagem disponível em [19].

VII. Base de Conhecimento

Nesta aplicação pode-se entender como a principal base de conhecimento os pesos gerados pela rede neural na etapa de treinamento. Para realizar essa etapa foi necessário criar um dataset com imagens que continham os objetos (classes) que se deseja detectar, criar documentos de anotação com a posição desses objetos em cada imagem e treinar a rede neural.

A. *Dataset*

O dataset foi constituído de um total de 360 imagens. Parte destas imagens foram retiradas de um banco de dados aberto [24], outra parte foi gerada pelo próprio autor. O número de imagens é pequeno para se atingir um alto nível de precisão, mas suficiente para validação inicial do modelo. Para resultados mais satisfatórios é sugerido um mínimo de 2000 imagens por classe [25]. Foi tomada a devida atenção para se obter imagens variadas, com diferentes níveis de luminosidade e resolução e com os objetos posicionados de diversas formas e em diferentes ambientes. A importância desse cuidado é a de diminuir a incidência de vieses na detecção.

B. *Anotação de imagens*

Com o auxílio do software de anotações labelImg [26], as imagens receberam etiquetas com as coordenadas dos objetos que se deseja detectar. As classes foram definidas de zero a seis, correspondendo respectivamente à: máscara, capacete, óculos, abafador, colete, luva e bota. A Fig. 17 ilustra a tela principal do software de anotações. Nele os objetos que se deseja detectar são marcados com uma caixa delimitadora. Em seguida é gerado um arquivo .txt com o mesmo nome da imagem. Em cada linha do arquivo será gravado, respectivamente: o número da classe, as coordenadas x e y do centro da caixa que delimita o objeto e a largura e a altura da caixa que está limitando o objeto. Com exceção do valor referente a classe, todos os outros são números entre 0 e 1. O arquivo gerado pode ser conferido na Fig. 18.

Fig. 17 Print de tela, gerada pelo autor, do software de anotações labelImg [26]. No exemplo, estão sendo marcadas a posição de dois capacetes e duas luvas de uma imagem adquirida de um banco de dados aberto [24].

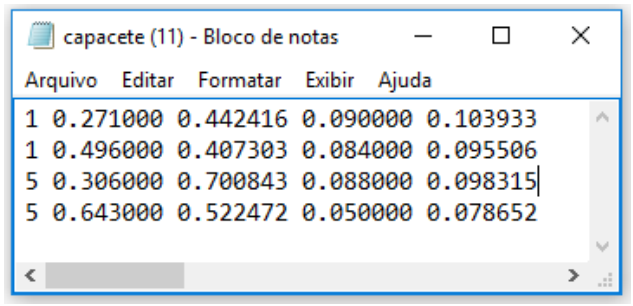

Fig. 18 Print de tela, gerado pelo autor, de arquivo .txt derivado da Fig. 16. Cada linha do arquivo se refere respectivamente à: classe, coordenada x, coordenada y, largura e altura da caixa que está delimitando o objeto.

Cada imagem utilizada continha no mínimo um objeto detectável. No total foram 2637 EPIs etiquetados. Sendo:

- 820 máscaras
- 507 capacetes
- 388 óculos
- 55 abafadores
- 90 coletes
- 264 luvas
- 513 botas

Para facilitar a aquisição de imagens, inicialmente a classe correspondente aos óculos também recebeu como positivo óculos de grau comuns, não somente óculos de proteção como seria o ideal. Da mesma forma a classe das botas também recebeu outros calçados como sapatos e tênis, e não somente botas e botinas de proteção como também seria o ideal.

Para melhoria do modelo no futuro, sugere-se equilibrar a quantidade de positivos em cada classe além de adicionar imagens sem objetos detectáveis junto de arquivos .txt vazios de mesmo nome. Essa prática pode melhorar o desempenho do treinamento da rede neural [25].

C. Etapa de treinamento

O treinamento foi realizado em ambiente virtual do Google Colab® em uma GPU Nvidea® Tesla K80. O espaço de memória alocado para o processo foi de 12Gb. Das 360 imagens utilizadas, 300 foram usadas como entrada e 60 imagens foram usadas na validação do modelo (20% das imagens de treinamento). Foram realizadas 3000 iterações em um tempo de aproximadamente 10 horas. Sugere-se para melhor desempenho do modelo que se aumente o tempo de treinamento até que a perda média deixe de diminuir [25] .

O treinamento utilizou o auxílio de uma rede pré-treinada. A intenção dessa técnica é diminuir o tempo de treinamento, pois a rede pré-treinada já aprendeu características comuns para diversos objetos nas primeiras camadas convolucionais.

Antes de iniciar o treinamento foram feitas mais algumas alterações na rede: primeiro foi definido a quantidade de imagens por lote que será aplicado a normalização (batch) em 64 e as subdivisões em 16, esses parâmetros influenciam na memória do hardware; em seguida o número máximo de iterações foi definido em 14000 conforme (14); por fim os passos foram definidos em 80% e 90% do $max\_batches$, respectivamente 11200 e 12600 [25] [27] [28].

$$\text{max\_batches} = 2000 \bullet classes \tag{14}$$

A métrica de avaliação do treinamento é mAP (métrica de precisão média). Essa métrica leva em consideração três parâmetros: IoU, $Precisão$ e $Recall$.

A intersecção sobre a união já foi descrita em (6), a métrica de precisão indica a incidência de falsos positivos e o recall indica se não houve falsos negativos. Em (15) e (16), A se refere a área do objeto e B a área da detecção.

$$Precisão = \frac{A \cap B}{A} \tag{15}$$

$$Recall = \frac{A \cap B}{B} \tag{16}$$

No final do treinamento as classes obtiveram os seguintes valores para mAP:

- Máscara: 94,36%
- Capacete: 75,07%
- Óculos: 77,67%
- Abafadores: 55,36%
- Coletes: 100%
- Luvas: 78,17%
- Botas: 49,54%

Os resultados do treinamento estão diretamente ligados com a qualidade do dataset e com o tempo de treinamento. Previa-se que os EPIs que seriam mais facilmente detectados, pelo seu tamanho e visibilidade, seriam a máscara, o capacete e o colete. Observando os resultados do treinamento podemos dizer que:

- A detecção da máscara obteve um excelente desempenho para o número de iterações.
- Acreditava-se que a detecção de capacetes teria um resultado semelhante ao da máscara, talvez o fato de que o dataset continha muitas imagens em que o EPI estava relativamente distante (por consequência muito pequeno na imagem) tenha afetado a qualidade da detecção.

- O aparente ótimo desempenho da detecção do colete pode estar relacionado a pouca variabilidade das imagens que continham este objeto.
- Já se acreditava que a detecção dos óculos, às vezes não tão visíveis devido a transparência, não atingiria alto desempenho tão rapidamente. Certamente a qualidade da detecção foi impulsionado pelo fato de que diversos tipos de óculos foram utilizados no treinamento, não só óculos de proteção.
- O baixo desempenho da detecção de abafadores certamente está ligado ao dataset precário desta classe. A principal dificuldade na detecção deste EPI está no fato de que ele envolve toda a cabeça, assim sua área de detecção coincide com a de outros EPIs, além de que a parte do objeto correspondente a haste pode ficar oculta em torno da cabeça em alguns casos, dificultando a rede de apreender padrões de continuidade.
- A incidência de falsos positivos na detecção de luvas e botas é muito provável. Certamente esse fator afetou a qualidade da detecção.

VIII. Reconhecimento e Interpretação

É denominado reconhecimento o processo de atribuição de um rótulo a um objeto baseado em suas características e de interpretação o processo de atribuir significado a um conjunto de objetos reconhecidos [8].

A partir de que uma nova imagem passe pela CNN, a rede retorna as prováveis regiões em que um objeto pode estar localizado e a probabilidade de cada objeto estar nessa região. É definido um percentual de confiança mínima para que esse EPI seja marcado. Usando as coordenadas de localização do objeto, é desenhada na imagem uma caixa delimitando sua área mínima, o nome da classe que o corresponde e a probabilidade da detecção. As coordenadas são comparadas com os landmarks, conforme descritos no item V.B. Se a região em que um determinado EPI foi detectado coincide com a ROI deste EPI, é considerado uma detecção positiva.

Na Fig. 19 pode ser conferido um teste de detecção de: máscara (99%), capacete (99%), óculos (51%), abafador (81%), colete (82%), luvas (ambas 100%) e botas (99% e 100%).

Fig. 19 Exemplo de detecção dos EPIs. Imagem gerada pelo autor.

IX. Prototipação

A maioria das aplicações modernas envolvendo visão computacional e CNNs são desenvolvidas para GPUs (Graphics Processing Unit) ou TPUs (Tensor Processing Unit). Esses são hardwares dedicados com ótimo desempenho em tempo real. Porém, visando um sistema de baixo custo, foi desenvolvido um protótipo em Python para ambiente Windows® que rodasse diretamente em uma CPU (Central Processing Unit) convencional as técnicas descritas até aqui.

A plataforma de desenvolvimento do protótipo foi composta de:

- **Sistema operacional:** Windows® 10 PRO 2016
- **Tipo de sistema:** 64 bits
- **Computador:** Acer® (Aspire E1-531)
- **Processador:** Intel® Celeron® CPU $1000M$ @ $1.80GHz$
- **Memória RAM:** $4Gb$
- **Câmera:** Webcam Padrão HD ($1280p \times 720p$)
- **Linguagem de programação:** Python 3.7
- **Ambiente de programação:** Pycharm® Community Edition 2021
- **Bibliotecas:** OpenCV (4.5.3.56), MediaPipe (0.8.6.2), tkinter, PIL, imutils, numpy, math e matplotlib.

Foi desenvolvido um programa que permite a visualização da câmera e contém um menu com os status dos EPIs. Quando se está considerando a detecção de um EPI o ícone respectivo a este equipamento fica em azul, no menu de configurações é possível mudar este status. Assim, quando o ícone referente a um determinado EPI está cinza, este equipamento não será considerado na detecção. Quando um objeto é detectado dentro de sua ROI, o ícone referente a este objeto fica verde. Quando o objeto é detectado, mas não coincide com a ROI, o ícone fica amarelo. O índice de detecção do EPI sempre é mostrado abaixo do ícone. A mensagem de acesso é mostrada na parte inferior. No botão 'Histórico' é possível acessar a pasta com as amostras positivas e negativas dos processos. Na Fig. 20 é possível visualizar a tela inicial do programa.

Fig. 20 Print de tela do protótipo. Imagem gerada pelo autor.

Na Fig. 21 é apresentado o fluxograma geral do protótipo, contendo seis etapas: inicialização, captura de imagem, estimativa de postura, detecção de objetos, comparação com zonas de interesse e tomada de decisão.

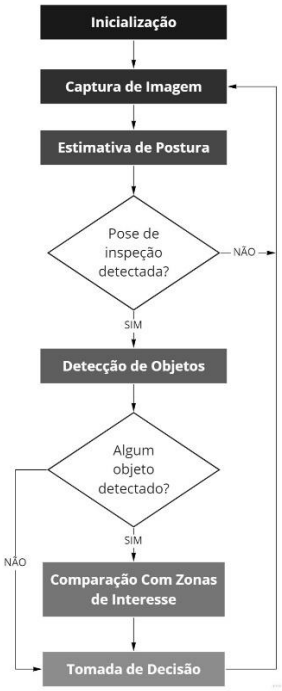

Fig. 21 Fluxograma geral do protótipo desenvolvido. Gerado pelo autor.

A seguir cada etapa é detalhada:

A. *Inicialização*

Essa é a etapa responsável por iniciar o programa, importando as bibliotecas, declarando as varáveis, lendo os arquivos de configuração da CNN e os pesos pré-treinados assim como configurando o backbone que será utilizado. Foi utilizado o módulo para redes neurais OpenCV-dnn para configuração darknet [29].

Em seguida é ativada a câmera que irá supervisionar o sistema e é aberta a janela do programa. Caso o programa não consiga ativar a câmera é mostrada a mensagem "Câmera não detectada", caso contrário é capturado o primeiro frame.

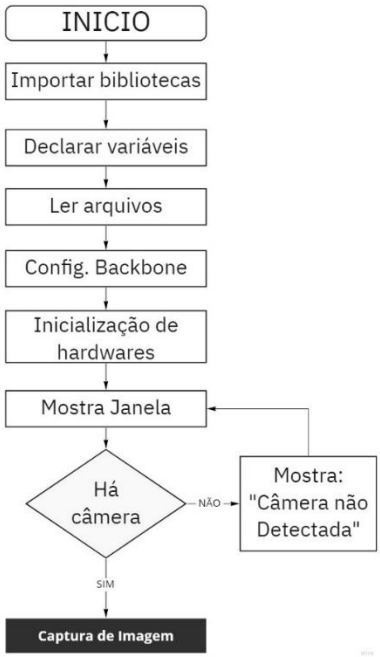

Fig. 22 Fluxograma detalhado da etapa de inicialização. Gerado pelo autor.

B. *Captura de imagem*

O frame capturado é redimensionado para $920p$, então é enviado para classe de estimativa de postura. O algoritmo detector tentará encontrar uma face humana, caso não encontre será capturado um novo frame, caso contrário iniciará o processo de estimativa de postura.

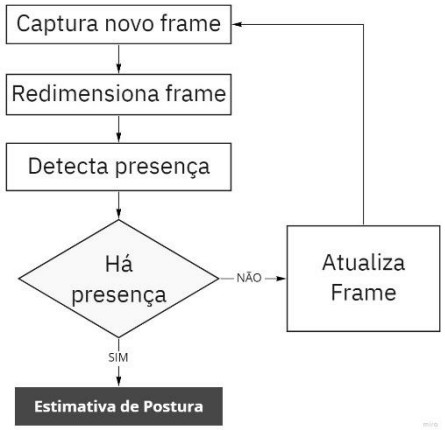

Fig. 23 Fluxograma detalhado da etapa de captura de imagem. Gerado pelo autor.

C. *Estimativa de postura*

Identificando os landmarks do corpo, procura-se calcular o ângulo dos braços utilizando (1). Se a postura correta for detectada, se a última detecção ocorreu a mais de cinco segundos e se a postura foi mantida por pelo menos três segundos, iniciará o processo de detecção de objetos nesse frame. Caso qualquer uma dessas condições não seja respeitada, um novo frame será capturado.

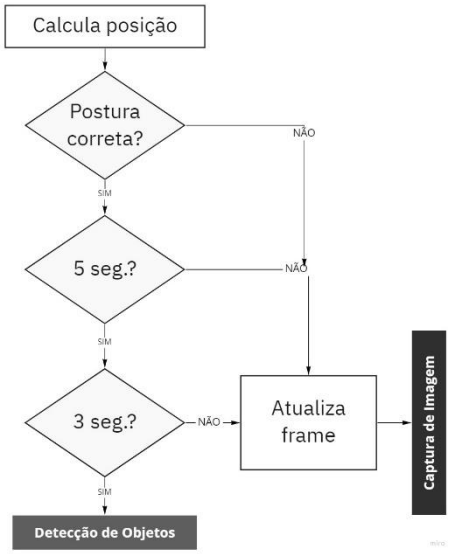

Fig. 24 Fluxograma detalhado da etapa de estimativa de postura. Gerado pelo autor.

D. *Detecção de Objetos*

O frame é convertido em um objeto BLOB e enviado para a primeira camada convolucional. Na saída da CNN retorna as varáveis de previsão. Caso o EPI seja detectado com uma confiança mínima, o frame é salvo em uma pasta de positivos junto com um arquivo .txt de mesmo nome com as coordenadas de localização do objeto na imagem. Caso nenhum EPI seja detectado com uma confiança

mínima, o frame será salvo em uma pasta de negativos junto de um arquivo .txt vazio. Posteriormente esses arquivos podem ser utilizados para retreinar a rede.

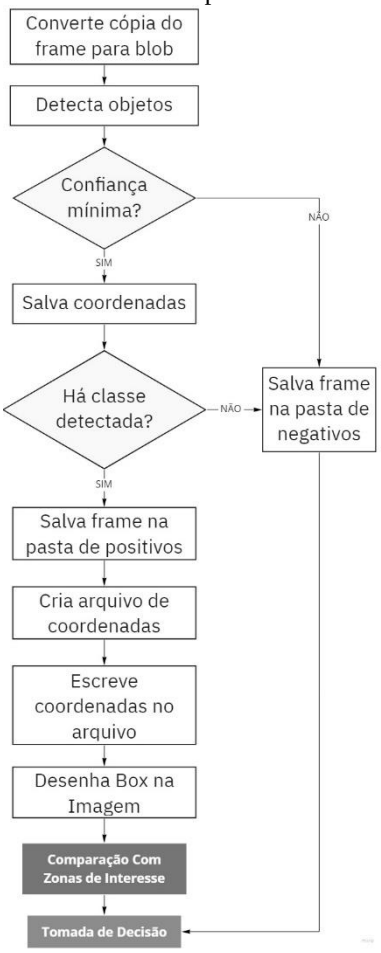

Fig. 25 Fluxograma detalhado da etapa de detecção de objetos. Gerado pelo autor.

E. Comparação com zonas de interesse

Nessa etapa, as coordenadas do EPI detectado serão comparadas com as coordenadas da ROI. Caso a coordenada do objeto não conhecida com a ROI, irá se considerar que o EPI está mal posicionado.

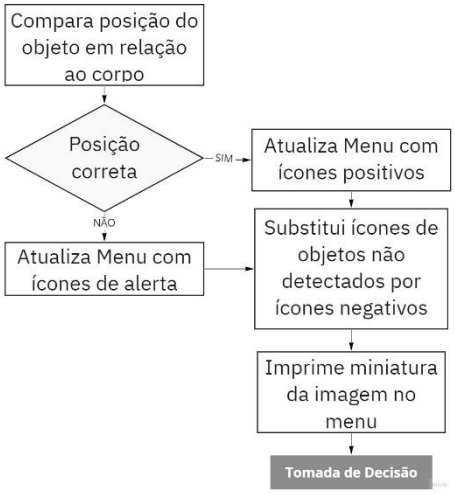

Fig. 26 Fluxograma detalhado da etapa de comparação com zonas de interesse. Gerado pelo autor.

F. Tomada de decisão

Se os EPIs foram detectados e sua posição coincida com a ROI, será mostrada uma mensagem de "Acesso Liberado". Caso a posição de algum EPI não coincida, será mostrado uma mensagem de "EPI Mal Posicionado". Caso algum EPI não tenha sido detectado, é mostrada a mensagem "Acesso Negado".

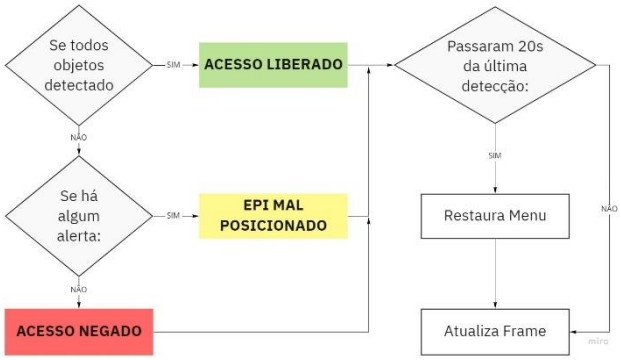

Fig. 27 Fluxograma detalhado da etapa de saída do sistema. Gerado pelo autor.

X. RESULTADOS PRÁTICOS

Para aferir os resultados, foram utilizados conceitos de métrica de precisão e de recall um pouco diferentes dos abordados no item VII.C.

Foram analisadas um total de 105 imagens, com diferentes EPIs, dispostos em diferentes posicionamentos e distâncias. Foram levadas em consideração as detecções: verdadeiro positivo (VP), falso positivo (FP) e falso negativo (FN). Não foi levado em consideração a área de intersecção da caixa que delimita o objeto, apenas se foi realizada a detecção ou não. A confiança mínima de detecção foi de 80% e a supressão não máxima de 30%.

$$Precisão = \frac{VP}{VP+FP} \quad (17)$$

$$Recall = \frac{VP}{VP+FN} \quad (18)$$

Os resultados podem ser conferidos na Tabela I.

TABELA I . AFERIÇÃO DO PROTÓTIPO

EPI	VP	FP	FN	Precisão	Recall
Máscara	30	3	0	90,91%	100%
Capacete	29	3	1	90,63%	96,67%
Óculos	23	0	7	100%	76,67%
Abafador	14	0	16	100%	46,67%
Colete	25	0	0	100%	100%
Luva Direita	16	0	7	100%	69,57%
Luva Esquerda	16	0	6	100%	72,73%
Bota Direita	10	0	0	100%	100%
Bota Esquerda	12	0	1	100%	92,31%
Total	175	6	38	96,69%	82,16%

XI. TRABALHOS FUTUROS

Algumas ideias para trabalhos futuros são:

- Aplicar processamento de imagem adaptativo, para que mudanças de iluminação afetem menos o modelo.
- Aumentar o dataset e o número de iterações da rede neural.
- Aplicar a técnica com todos EPIs presentes na NR6.
- Embarcar o sistema em dispositivos como Raspberry Pi ou NVIDEA Jetson Nano.

XII. QUESTÕES ÉTICAS EM VISÃO COMPUTACIONAL

A lei brasileira permite o monitoramento do ambiente de trabalho com a finalidade de auxiliar no acompanhamento de tarefas laborais ou fiscalizar eventuais condutas do colaborador. Porém não é permitido que se viole os direitos à intimidade e privacidade. Ou seja, é imprescindível que sistemas similares ao sistema abordado nesse artigo sejam aplicados apenas em locais que tenham o objetivo da proteção e segurança das pessoas e que aja o consentimento de todos os envolvidos. Esses sistemas não devem substituir o treinamento e a conscientização dos trabalhadores, mas serem potenciais auxiliadores do bem estar no ambiente de trabalho.

XIII. CONCLUSÃO

O modelo foi capaz de detectar EPIs a uma distância de até três metros em 82% dos casos com uma taxa de precisão de 96,7%. Os equipamentos mais facilmente detectados foram máscaras e capacetes. O EPI que o protótipo possui maior dificuldade de detectar é o abafador. A qualidade do banco de imagens deste objeto está diretamente ligada a esse desempenho. Nos testes utilizando o método de estimativa de postura foi possível melhorar a qualidade da detecção, diminuindo consideravelmente a incidência de falsos positivos. O protótipo também se mostrou útil no processo de formação de um dataset próprio, que pode ser usado para melhorar o modelo no futuro. O projeto comprovou a possibilidade de usar visão computacional e redes neurais para auxiliar em processos de segurança do trabalho.

Todos os códigos do projeto estão disponíveis em: https://github.com/felipeSperb/VCAD_EPIs.

XIV. REFERÊNCIAS

[1] M. P. d. Trabalho, "Plataforma Smartlab de Trabalho Decente," Smartlab, [Online]. Available: https://smartlabbr.org. [Acesso em 5 Setembro 2021].

[2] P. Basilio, "Brasil é 2º país do G20 em mortalidade por acidentes no trabalho," G1, 01 Maio 2021. [Online]. Available: https://g1.globo.com/economia/noticia/2021/05/01/brasil-e-2o-pais-do-g20-em-mortalidade-por-acidentes-no-trabalho.ghtml. [Acesso em 20 Outubro 2021].

[3] A. Z. F. e. S. A. d. Nascimento, "Anuário Estatistico de Acidentes do Trabalho - AEAT," Ministério da Fazenda, Instituto Nacional de Seguro Social, Empresa de Tecnologia e informações da Previdência, Brasília - DF, 2018.

[4] *NR 6 - Equipamentos de proteção individual - EPI, Portaria GM n3.214 de 8 de junho de 1978.*

[5] D. J. Simons e C. F. Chabris, "Gorillas in our midst: sustained inattentional blindness for dynamic events," 20 Junho 1999.

[6] R. A. Rensink, J. K. O'Regan e J. J. Clark, "To see or not see: the need for attention to perceive changes in scenes," *American Psychological Society,* vol. 8, nº 5, pp. 368-373, Setembro 1997.

[7] R. C. Gonzalez e R. E. Woods, Digital Image Processing, New Jersey: Pearson Education, 2008.

[8] O. M. Filho e H. V. Neto, Processamento Digital de Imagens, Rio de Janeiro: Brasport, 1999.

[9] "WF EPI," WF Equipamentos de Proteção LTDA., [Online]. Available: https://www.wfepi.com.br/. [Acesso em 1 Novembro 2021].

[10] "OpenCV Tutorial," [Online]. Available: https://opencv-tutorial.readthedocs.io/. [Acesso em 02 Novembro 2021].

[11] "MediaPipe," Google, [Online]. Available: https://google.github.io/mediapipe/solutions/pose. [Acesso em 05 Novembro 2021].

[12] "Google AI Blog," Google, [Online]. Available: https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html. [Acesso em 05 Novembro 2021].

[13] I. Goodfellow, Y. Bengio e A. Courville, Deep Learning, MIT Press, 2016.

[14] Y. Lecun, Y. Bengio e P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings os the IEEE,* vol. 86, nº 11, pp. 2278-2324, Novembro 1998.

[15] A. Krizhevsky, I. Sutskever e G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks".

[16] Viceri, "Entendendo de vez a convolução: base para processamento de imagens," Viceri, 2019. [Online]. Available: https://viceri.com.br/insights/entendendo-de-vez-a-convolucao-base-para-processamento-de-imagens/. [Acesso em 1 Novembro 2021].

[17] J. Redmon, S. Divvala, R. Girshick e A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv,* 2016.

[18] J. Redmon e A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv,* 25 Dezembro 2016.

[19] J. Redmon e A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv,* 2018.

[20] A. Bochkovskiy, C.-Y. Wang e H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv:2004.10934v1,* 23 Abril 2020.

[21] S. Ioffe e C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," *arXiv:1502.03167,* 2 Março 2015.

[22] "Data Science Academy. Deep Learning Book," 2021. [Online]. Available: https://www.deeplearningbook.com.br/. [Acesso em 10 Novembro 2021].

[23] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye e D. Ren, "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression," *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20),* 2020.

[24] "Roboflow," [Online]. Available: https://public.roboflow.com/. [Acesso em 10 Agosto 2021].

[25] A. Boshkovsk, "darknet," GitHub, [Online]. Available: https://github.com/AlexeyAB/darknet. [Acesso em 2021 Novembro 18].

[26] D. Tzutalin, "labelImg," GitHub, [Online]. Available: https://github.com/tzutalin/labelImg. [Acesso em 18 Novembro 2021].

[27] A. Boshkovsk, "Parâmetros CFG na seção [net]," GitHub, [Online]. Available: https://github.com/AlexeyAB/darknet/wiki/CFG-Parameters-in-the-%5Bnet%5D-section. [Acesso em 25 Novembro 2021].

[28] A. Boshkovsk, "CFG Parameters in the different layers," GitHub, [Online]. Available: https://github.com/AlexeyAB/darknet/wiki/CFG-Parameters-in-the-different-layers. [Acesso em 25 Novembro 2021].

[29] O. O. S. C. Vision, "Deep Neural Network module," [Online]. Available: https://docs.opencv.org/3.4/d6/d0f/group__dnn.html. [Acesso em 30 Agosto 2021].