CLASSIFICAÇÃO DE IMAGENS DE PESSOAS QUANTO A ESTAREM UTILIZANDO MÁSCARA POR REDES CONVOLUCIONAIS EM SISTEMA EMBARCADO

Fernando Marcel Cardozo ÉVORA[1]
IFSP/Câmpus São Paulo

Vinícius de Barros SOUSA[2]
IFSP/Câmpus São Paulo

Ricardo PIRES[3]
Doutor em Sistemas Automáticos e Microeletrônicos/Université Montpellier II
Docente do Departamento de Elétrica
IFSP/Câmpus São Paulo

RESUMO
Este artigo apresenta o desenvolvimento de um sistema embarcado com uma câmera de vídeo o qual, a partir da imagem obtida desta, visa a classificar as imagens de indivíduos em dois grupos, "Com máscara" ou "Sem máscara". Para a classificação da imagem do indivíduo, foram utilizados algoritmos de inteligência artificial, sendo o algoritmo principal baseado em Redes Neurais Artificiais Convolucionais. A rede neural convolucional criada teve uma acurácia de 99% para o conjunto de validação e classificou apenas dois casos como falsos positivos para um novo conjunto de 200 novos exemplos. Em operação, este sistema exibe uma taxa de quadros classificados de cerca de 5 quadros/s. Para a identificação das faces a serem classificadas, o sistema exige que o ambiente tenha iluminação adequada. Este sistema destina-se a ser utilizado durante pandemias como a do COVID-19, como uma ferramenta de controle da propagação deste vírus.
Palavras-chave: COVID-19; Redes Neurais; Máscara; Pandemia; Classificação de Imagens.

Introdução

O ano de 2020 pode ser considerado como o ano em que a humanidade viu a propagação de uma nova doença, a COVID-19, doença esta que culminou em uma pandemia e que se estende durante o ano de criação deste artigo, 2021, sendo registrados milhões de

[1] Endereço eletrônico: fernandocardozoo@hotmail.com
[2] Endereço eletrônico: vinicius.barros@aluno.ifsp.edu.br
[3] Endereço eletrônico: ricardo_pires@ifsp.edu.br

infectados e milhões de mortos (OMS, 2021, 1). O número de casos e vítimas desta doença aumenta progressivamente, sendo que, para controlar a sua disseminação, foram propostas medidas preventivas pela Organização Mundial de Saúde (OMS). Dentre estas medidas, uma das principais é o uso de máscaras em locais públicos e de confinamento (OMS, 2020, 1).

Máscaras devem ser usadas como uma parte de um conjunto de medidas do tipo "*Do it all*" (Sigam todas as medidas), onde, além do uso de máscaras, devem-se seguir outras medidas, bem como distanciamento físico, evitar aglomerações, evitar ambientes fechados, realizar higienização das mãos, cobrir espirros, tosse e muito mais (OMS, 2020, 1).

As teses seguintes podem ser consideradas para a relevância do uso de máscaras durante esta pandemia, como descrito no trecho do documento de diretrizes provisórias da Organização Mundial de Saúde, onde um pequeno estudo de coorte de Pequim descobriu que o uso de máscaras, pelos membros de uma família sem nenhum membro infectado, foi 79% eficaz na redução da transmissão da doença. Ainda de acordo com o documento, um estudo de caso-controle da Tailândia descobriu que usar uma máscara durante o contato com pacientes infectados pelo COVID-19 estava associado a uma diminuição de 77% no risco de infecção (OMS, 2020, 8).

Tendo em vista a medida preventiva do uso de máscaras, o presente artigo apresenta a proposta de criação de um sistema embarcado, que por meio de uma câmera de vídeo e da imagem obtida a partir dela, tem como objetivo classificar imagens de indivíduos em dois grupos, "Com máscara" ou "Sem máscara". Este sistema visa a ser utilizado durante pandemias como a da COVID-19, podendo ser uma ferramenta de controle de disseminação desta doença ou similares.

A identificação de faces em imagens e suas respectivas classificações quanto ao uso de máscaras são um dos principais desafios deste projeto. Para este problema, podem-se considerar algumas soluções, bem como a solução apresentada em (CHOWDARY *et al.*, 2020, 1), no qual foi empregado o método de transferência de aprendizagem usando como base um modelo de rede neural artificial convolucional denominada *InceptionV3* (SZEGEDY *et al.*, 2016, 2822). O método de transferência de aprendizagem, do inglês *Transfer Learning*, consiste em utilizar e "retreinar" um modelo de rede neural artificial previamente definido e/ou treinado para tarefas de classificação ou regressão que não necessitam ser as mesmas para as quais foram concebidas originalmente. Além deste, o método de transferência de aprendizagem também foi utilizado em (LOEY *et al.*, 2021, 1) para o problema de classificação de imagens de indivíduos quanto ao uso de máscara. No entanto, o modelo de rede neural convolucional utilizada como base para a extração de características foi o *ResNet50* (HE *et al.*, 2016, 775). Ambos os modelos, *InceptionV3* e *ResNet50*, foram originalmente concebidos tendo como tarefa a classificação de

imagens em um conjunto de mil classes possíveis. Além deste fato, trouxeram contribuições para o problema de "desaparecimento do gradiente", do inglês *"Vanishing Gradient Problem"* (HOCHREITER, 1997, 2). O problema de "desaparecimento do gradiente" surge em redes neurais artificiais cujo aprendizado é do tipo supervisionado e baseado no algoritmo de *retropropagação* pela "descida estocástica do gradiente", tendo correlação direta com o aumento do número de camadas concatenadas de parâmetros treináveis.

A solução do problema de classificação de imagens de indivíduos quanto ao uso de máscara proposta para o desenvolvimento do sistema embarcado citado neste artigo não se baseia no método de transferência de aprendizagem, mas sim no desenvolvimento de um modelo de rede neural artificial convolucional clássica operando em conjunto com o algoritmo de detecção de faces baseado no algoritmo classificador "*Haar Cascade"* (JONES e VIOLA, 2001, 1). Os motivos para não ser utilizado o método de transferência de aprendizado tendo como base os modelos *InceptionV3* e *ResNet50* e/ou similares, devem-se ao fato de que a grande maioria destes modelos apresentam um custo computacional elevado, pois são idealizados para a classificação de imagens em um grande número de classes, especificamente mil classes distintas para ambos, sendo inadequado empregá-los para a tarefa de classificação de imagens de menor complexidade como na classificação de apenas duas classes, sendo elas "Com máscara" e "Sem máscara". O modelo de rede neural convolucional proposta neste artigo, possui apenas 2 camadas convolucionais, em contrapartida às 50 camadas convolucionais em um modelo *ResNet50* e às 48 camadas convolucionais em um modelo *InceptionV3*.

Fundamentação teórica

Por imagem, em resumo pode-se compreender uma combinação de cores representadas em um espaço bidimensional, onde cada cor corresponde a um comprimento de onda no espectro eletromagnético da luz visível. Uma imagem digital, por sua vez, pode ser compreendida como uma representação digital para uma imagem real, onde é realizada a amostragem e quantização de uma imagem, através do uso de sensores/transdutores que geram tensões de saída de acordo com a amplitude e frequência (ou comprimento de onda) da onda eletromagnética refletida por um objeto. (GONZALEZ e WOODS, 1992, 52).

Uma imagem contínua pode ser representada em um plano bidimensional f(x,y). Uma imagem deve ser contínua com respeito às coordenadas x- e y-, e também em amplitude. Para convertê-la para a forma digital, deve-se amostrar a função contínua f(x,y) tanto em coordenadas como em amplitude. A digitalização dos valores das coordenadas é chamada de

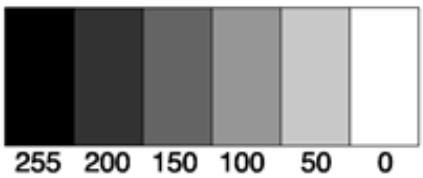
amostragem. A digitalização das amplitudes nestas coordenadas é chamada de *quantização*. (GONZALEZ e WOODS, 1992, 52).

O resultado da amostragem e quantização de uma imagem resulta em uma matriz de números reais. Assumindo que uma imagem f(x,y) é amostrada e quantizada, pode-se representar uma imagem digital resultante como uma matriz contendo *m* linhas e *n* colunas. Os valores das coordenadas (x, y) tornam-se quantidades discretas. Para uma notação clara e conveniente, devem-se usar apenas valores inteiros para estas coordenadas discretas. O valor para as coordenadas na origem são (x, y) = (0,0). (GONZALEZ e WOODS, 1992, 60).

Para cada coordenada *m* x *n*, de uma matriz de uma imagem, pode-se denominar esta coordenada como sendo equivalente a um *"pixel"*. É comum, em se tratando de representações digitais de imagens, uma imagem ser representada por uma única matriz, contendo *m* linhas e *n* colunas, onde os valores para as coordenadas *m* x *n* podem assumir valores binários, do tipo 0 ou 1, ou então valores inteiros entre 0 e 255.

Para as imagens que assumem valores binários, serão visualizadas imagens em escala de preto e branco, sendo que o valor 0 representa a cor branca e o valor 1 representa a cor preta. Já para as imagens que assumem os valores inteiros de 0 a 255, serão visualizadas imagens em escalas de cinza, sendo que o valor 0 representa a cor branca, o valor intermediário de 128 representa a cor cinza considerada padrão e o valor 255 representa a cor preta. A faixa de valores, que vão de 0 a 255, são equivalentes a uma representação de 8-bits para cada pixel, já que $2^8 = 256$. Isso se deve ao fato de 8-bits ser uma quantidade de bits convencional para registradores utilizados em micro-processadores/controladores de baixo custo. Atualmente, um grande número de dispositivos utilizam representações contendo 32-bits de informação por pixel ou mais. No entanto a representação digital de imagens neste trabalho tem como foco a representação do tipo escalas de cinza e *R.G.B ou B.G.R.* A figura 01 ilustra a representação do padrão de cor escalas de cinza e a figura 02 mostra uma imagem representada em escalas de cinza e em escala de preto e branco.

Figura 1: Imagem representando as cores e seus respectivos valores para o padrão de escalas de cinza



Fonte: Autores

Figura 2: Diferença de representação de cor para uma mesma imagem. Em escalas de cinza (à esquerda) e binária (à direita)

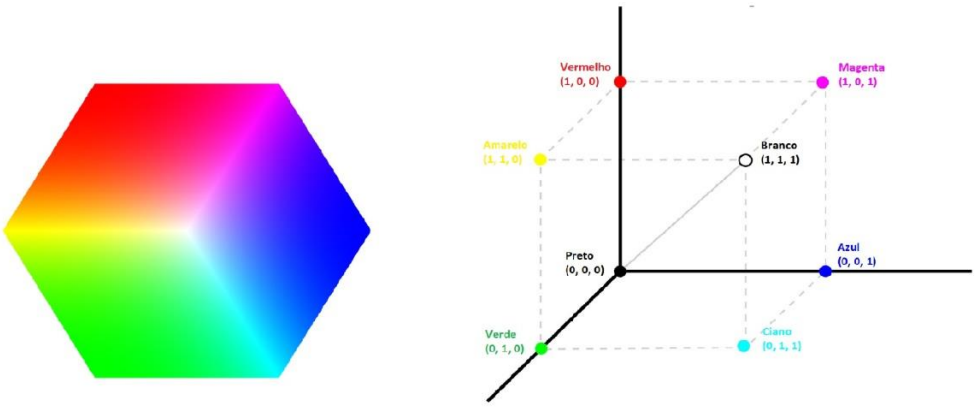
Fonte: Alessio Damato, licença *Creative Commons*

Uma cor pode ser representada por um vetor de três dimensões. O conjunto de todas as cores forma um espaço vetorial chamado espaço de cores ou modelo de representação de cor. As três componentes de uma cor podem ser definidas de diversas formas, levando a vários modelos de representação de cor. (PLATANIOTIS e VENETSANOPOULOS, 2000, 1).

Dentre os espaços de representação de cores existentes, o mais utilizado em sistemas de processamento de imagens, como computadores gráficos e sistemas de multimídia, é o RGB. A representação RGB consiste em reproduzir qualquer cor através da combinação de três cores fundamentais, sendo elas: Vermelho, Verde e Azul, em inglês, Red, Green and Blue, que dá origem ao acrônimo RGB.

Uma imagem digital colorida é representada por uma matriz bidimensional de vetores de três variáveis, os quais se referem aos valores vermelho, verde e azul de um pixel. (PLATANIOTIS e VENETSANOPOULOS, 2000, 11).

Figura 3: Modelo de representação RGB



Fonte: Autores

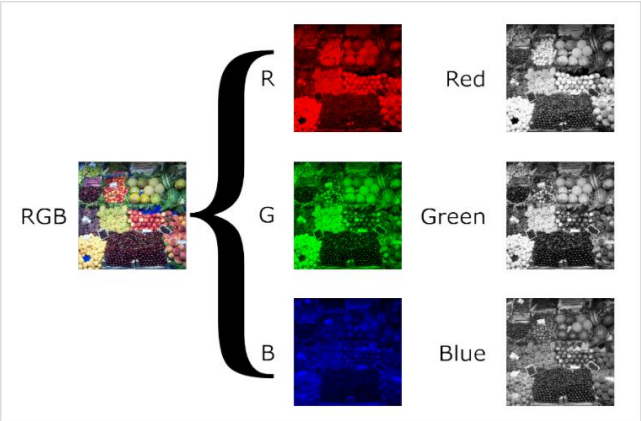
Na figura 03, pode-se observar que para cada cor no espaço RGB, existe uma combinação das três componentes, sendo que o vetor [0, 0, 1] corresponde à cor azul, o vetor [1,

0, 0] corresponde à cor vermelho e o vetor [0, 1, 0] corresponde à cor verde. As demais cores podem ser compreendidas como uma combinação entre os três vetores.

O padrão de representação RGB24 é equivalente ao padrão RGB para representação de cores para cada pixel de uma imagem, porém para imagens digitais, obtidas após amostragem e quantização de imagens reais. É comum imagens serem representadas em sistemas de processamento digital de imagem, como uma combinação de três matrizes ou canais RGB.

Ou seja, imagens podem ser representadas em sistemas de processamento digital de imagens, como sendo uma combinação de três matrizes, representando cada matriz um canal: R vermelho, G verde e B azul. Cada uma das três matrizes deve ser da mesma dimensão *m* x *n*, sendo que cada ponto dessas matrizes deve ter valores inteiros entre 0 a 255, assim como para as imagens em escala de cinza, têm-se $2^8 = 256$ valores ou intensidades possíveis para cada ponto em cada canal. Desta maneira, cada pixel da imagem será composto pela combinação de pontos *m* x *n* de cada canal, podendo ser considerado como um vetor tridimensional, cujas componentes são $[Valor(m \text{ x } n)^{\mathbf{R}}; Valor(m \text{ x } n)^{\mathbf{G}}; Valor(m \text{ x } n)^{\mathbf{B}}]$ = ***Pixel (M x N)***. Pode-se observar na figura 04 a representação dos três canais RGB para uma imagem colorida.

Figura 4: Representando a divisão em 3 canais RGB para uma imagem e cada canal representado em escala de cinza



Fonte: Nevit Diman, licença *Creative Commons.*

Para que seja possível extrair informações contidas em imagens, torna-se por vezes necessário realizar transformações sobre ela. Os tipos de transformações utilizadas para o desenvolvimento deste projeto foram corte, redimensionamento e conversão de modelo de cores RGB para escala de cinza.

Uma imagem digital pode ser compreendida como a combinação de uma ou mais matrizes de mesma dimensão, onde cada ponto da matriz é resultado de uma quantização e amostragem de uma imagem real. Assim sendo, para que seja feito o corte em uma imagem, basta identificar e delimitar as coordenadas ou pixels desejados e, em seguida, excluir todos os

pixels restantes, formando então uma nova imagem. Em geral e no caso deste trabalho, os cortes em imagens são delimitados por áreas retangulares com o objetivo de isolar faces na imagem.

Ou seja, por exemplo, para uma imagem de dimensão 224 linhas x 224 colunas, é especificada uma região para corte desta mesma imagem de área retangular de dimensão 112 linhas x 112 colunas, a partir de uma dada coordenada sobre a imagem original. Uma nova imagem resultante desta seleção terá a dimensão de 112 linhas x 112 colunas, o que implica em uma imagem de dimensão inferior a original. Ainda assim, pode-se entender a transformação de corte de uma imagem como sendo o resultado de uma operação lógica do tipo "AND" da matriz da área de corte especificada com a imagem original.

O resultado do corte de uma imagem leva sempre a uma imagem de área ou número de linhas e colunas inferior a imagem original. Como uma forma de contornar este fenômeno, pode-se realizar uma nova transformação sobre a imagem resultante chamada de Redimensionamento, neste caso, uma ampliação.

O redimensionamento ou mudança de escala de uma imagem consiste basicamente em aumentar ou diminuir o seu número de linhas e colunas, implicando em mais ou menos pixels. Este processo, apesar de parecer simples, exige, na maioria dos casos, que uma ou mais operações sejam realizadas, a depender se será feita a ampliação da imagem ou a sua redução.

Ao ampliar uma imagem, parte-se de uma imagem com um número inferior de linhas e colunas, em relação ao número de linhas e colunas da nova imagem que se deseja obter. Para isso, operações de preenchimento são necessárias, sendo comum o uso de interpolações. Dada uma imagem A[t] com *m* x *n*, linhas e colunas, a imagem ampliada A[t+1] com I x J, linhas e colunas, deve ter pelo menos, I > M ou J > N.

Ao se reduzir uma imagem, parte-se de uma imagem com um número superior de linhas e colunas, em relação ao número de linhas e colunas da nova imagem que se deseja obter. Para isso, operações para que não haja perda de informação são necessárias, sendo comum uma operação onde se calcula a média aritmética de uma janela dos valores contidos na(s) matriz(es) da imagem, para formar o valor das matrizes da imagem reduzida.

Neste trabalho, foi necessário transformar as imagens para o padrão de cor escalas de cinza, como uma etapa anterior ao algoritmo de identificação de faces *Haar Cascade.* Para transformar uma imagem RGB, ou seja, composta por três canais, ou matrizes, para uma imagem de um único canal em escala de cinza, é necessário realizar um método ponderativo, onde cada pixel da nova imagem será resultado da combinação ponderada de cada elemento do pixel RGB. A nova imagem, em escala de cinza, terá cada pixel preenchido da seguinte maneira pelo método conhecido por *Luminância*: imagem em escala de cinza (**C**) = [(0,3 **R**)+(0,59

G)+(0,11 **B**)], onde **R** é o canal Vermelho, **G** é o canal Verde e **B** é o canal azul (GARRISON e KANAN, 02, 2012).

Uma rede neural artificial convolucional RNC ou em inglês *CNN*, é um algoritmo de aprendizado de máquina, o qual recebe uma imagem de entrada, atribui importância a diversos aspectos ou objetos nesta imagem, por meio de pesos e vieses treináveis, e ainda é capaz de diferenciar um aspecto de outro. A arquitetura de uma rede RNC é análoga ao padrão de conexão entre os neurônios do cérebro humano, sendo inspirado na organização do córtex visual. A estrutura básica de uma RNC consiste em três tipos de camadas: camada convolucional, camada de *pooling* e uma camada *fully-connected* ou de classificação, sendo em geral uma camada de perceptron multicamadas. O treinamento de uma rede RNC ocorre pelo método de retropropagação. (LECUN, 07, 1998)

Em resumo, em uma camada convolucional, é realizada a operação de convolução entre um banco (conjunto) de filtros, também conhecido por *kernel*, e uma imagem de entrada. A sua entrada é uma matriz tridimensional com n_1 mapas de característica de dimensão $n_2 \, x \, n_3$. A saída é uma matriz tridimensional y, composta por m_1 mapas de características de dimensão $m_2 \, x \, m_3$. Um filtro treinável K_{ij} no banco de filtros tem dimensão l_1 x l_2 e conecta o mapa de característica de entrada x_i ao o mapa de característica de saída y_j. O módulo calcula a seguinte expressão (1):

$$y_j = b_j + \sum_i k_{ij} * x_i \tag{1}$$

onde $*$ representa o operador de convolução bidimensional discreta e b_j é um parâmetro, viés, treinável. Cada filtro detecta uma característica particular para cada localização na entrada. Usualmente, para redes neurais convolucionais clássicas, pode ser aplicada uma camada ou operação de não linearidade após a operação de convolução, denominada como função de ativação. Esta operação pode ser representada por funções matemáticas não-lineares, como a *Sigmoid*, *ReLu*, *Tanh* dentre outras. (FARABET, KAVUKCUOGLU e LECUN, 01, 2010)

A função de ativação, chamada de unidade linear retificada, do inglês *Rectified Linear Unit* ReLu, é geralmente utilizada como função de ativação durante a etapa convolucional. Essa função resulta em zero para qualquer valor negativo e cresce identicamente o seu resultado para valores positivos (CHEN et al., 02, 2015):

$$\text{ReLu}(x) = \{0, \text{ se } x < 0 \text{ e } x, \text{ se } x > 0\} \tag{2}$$

Um elemento importante da etapa convolucional é o deslocamento ou *stride*, que é realizado pelo filtro sobre uma matriz. O *stride* pode ser compreendido como quantas linhas e

colunas serão deslocadas pela matriz do filtro sobre a matriz da imagem, em cada etapa a se realizar a convolução.

Após uma camada convolucional, é concatenada uma camada de *Pooling.* Esta camada trata cada mapa de característica separadamente. Em sua forma mais simples, ela calcula a média dos valores sobre uma vizinhança (matriz) em cada mapa de característica. As vizinhanças são deslocadas por um *stride* de valor maior que um. Isto resulta em uma redução de dimensionalidade do mapa de características de saída. A operação de média é por vezes substituída por uma operação que seleciona valor máximo, denominada de *MaxPooling*. (FARABET, KAVUKCUOGLU e LECUN, 02, 2010)

Para que as matrizes resultantes da(s) camada(s) convolucional(is) possam ser passadas para a camada classificadora de Perceptron de Multicamadas, estas devem ser transformadas para que fiquem na forma vetorial e não mais na forma matricial. Existem diversos métodos para isto, como cálculo de valor médio nas matrizes, mas há também o método mais direto conhecido por *flatten* que consiste em concatenar as colunas das matrizes, de forma a "vetorizar" a informação.

As redes Perceptron de Múltiplas Camadas (PMC) são caracterizadas pela presença de, pelo menos, uma camada intermediária de neurônios situada entre a camada de entrada e a respectiva camada neural de saída. As redes PMC pertencem à arquitetura *feedforward* de camadas múltiplas, cujo treinamento é efetivado de forma supervisionada (SILVA, SPATTI e FLAUZINO, 2016, 91).

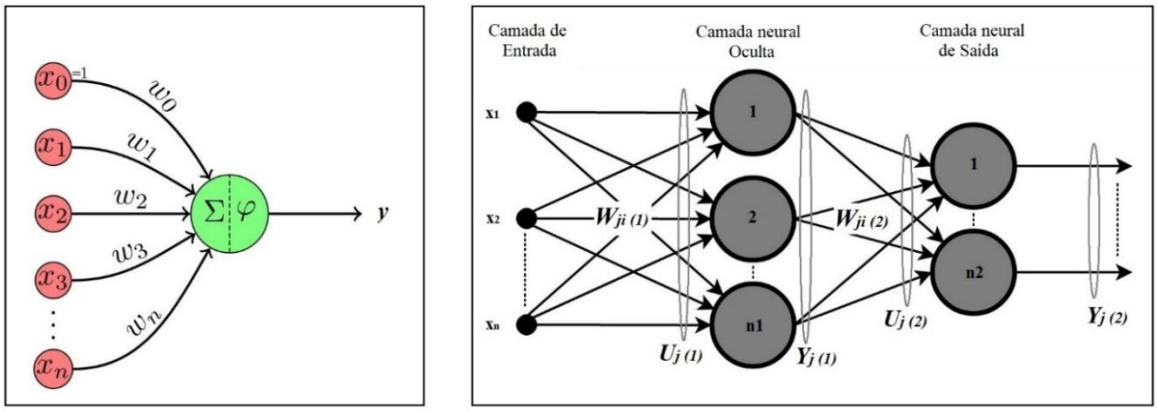
O processamento interno realizado por um único neurônio artificial Perceptron pode ser descrito pelas seguintes expressões (3) e (4):

$$u = \sum_{i=1}^{n} w_i . x_i - \theta \tag{3}$$

$$y = g(u) \tag{4}$$

Onde x_i são as entradas da rede, w_i é o peso associado a *i*-ésima entrada e θ é o limiar de ativação, $g(.)$ é a função de ativação, u é o potencial de ativação e y é a saída da rede. Pode-se visualizar, na figura 05, que o PMC é composto pela combinação de mais de um neurônio artificial Perceptron (SILVA, SPATTI e FLAUZINO, 2016, 52).

Figura 5: À esquerda um único Perceptron (Fonte: Martin Thomas, licença *Creative Commons*) e à direita uma rede PMC



Fonte: Autores

Em resumo, o algoritmo de treinamento *backpropagation* consiste em apresentar um conjunto de treinamento formado por valores de entradas e saídas à rede neural, de tal modo que cada amostra de treinamento apresentada à entrada da rede gere uma saída e a saída gerada pela rede será comparada com a saída esperada no conjunto de treinamento, por meio de uma função de custo ou erro. Partindo do valor obtido na função de custo ou erro, os pesos W da rede serão atualizados com o objetivo de diminuir o valor deste erro, até que a rede passe a apresentar valores de saída satisfatórios e que sejam iguais ou próximos do esperado.

Existem diversas funções de custo ou erro e diversos métodos de atualização dos pesos W da rede, não sendo possível abordar todos. Assim sendo, será dada uma breve explicação sobre os métodos utilizados para este projeto, sendo eles *Binary Cross Entropy* para a função de custo e *Stochastic Gradient Descent* para atualização dos pesos.

A função de custo *Binary Cross Entropy* padrão é representada por J_{bce}:

$$J_{bce} = -\frac{1}{M}\sum_{m=1}^{M}\left[y_m \times \log\left(h_\theta(x_m)\right) + (1 - y_m) \times \log\left(1 - h_\theta(x_m)\right)\right] \qquad (5)$$

Onde, M é o número de exemplos de treinamento, y_m é a saída esperada para a amostra de treinamento m, x_m é a entrada referente a amostra de treinamento m e h_θ é o modelo de rede neural com peso θ. (HO e WOOKEY, 2019, 2).

Considerando um modelo de aprendizado supervisionado, cada exemplo z é um par (x,y), composto de uma entrada x arbitrária e um escalar y de saída. Considera-se uma função de custo $l(\hat{y}, y)$, que mede o custo da predição de $\hat{y}$ quando a saída atual é y, e quando a família F de funções $f_w(x)$ é parametrizada por um vetor de pesos w. Busca-se a função $f \in F$ que minimize a função de $Q(z, w) = l(f_w(x), y)$ calculada sobre os exemplos. Seria desejável calcular sobre uma distribuição $dP(z)$ desconhecida. No entanto, frequentemente realiza-se o cálculo

sobre amostras $z_1 ... z_n$. O risco empírico e o risco esperado podem ser representados pelas expressões a seguir:

$$E(f) = \int l(f(x), y) dP(z) \quad (6)$$

$$E_n(f) = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i) \quad (7)$$

O risco empírico $E_n(f)$ mede o desempenho do conjunto de treinamento. O risco esperado $E(f)$ mede o desempenho de generalização, que é o desempenho esperado nos exemplos futuros. A teoria de aprendizagem estatística (CHERVONENKIS e VAPNIK,1971, 264) justifica minimizar o risco empírico, em vez do risco esperado, quando uma dada família F é suficientemente restritiva. (BOTTOU, 2012, 1)

O algoritmo *Stochastic Gradient Descent* (SGD), em português Gradiente Descendente Estocástico, ao invés de computar o gradiente de $E_n(f_w)$ diretamente, estima a cada iteração este gradiente, baseado em um único exemplo z_t randomicamente selecionado. Assim sendo, pode-se representar a atualização do vetor de pesos w_{t+1}, como sendo:

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t) \quad (8)$$

O processo estocástico $\{w_t, t=1, \ldots\}$ depende dos exemplos randomicamente escolhidos a cada iteração. O símbolo γ representa a taxa de aprendizagem, cujo valor é empiricamente atribuído (BOTTOU, 2012, 2).

Desenvolvimento do projeto

Para o desenvolvimento do projeto, foram estabelecidos objetivos e critérios, sendo o objetivo implementá-lo em um sistema embarcado capaz de identificar e classificar indivíduos que estejam ou não utilizando máscara, a partir de imagens obtidas por uma câmera de vídeo. A etapa de classificação seria realizada por uma rede neural convolucional.

Como critérios, o sistema embarcado deveria ter a capacidade de processar e manipular imagens em conjunto com um modelo de rede neural convolucional previamente treinado, ter um sistema operacional, executar instruções em linguagem de programação *Python*, suportar as bibliotecas necessárias, bem como *OpenCV* e *TensorflowLite* e, por fim, ter um baixo custo. O sistema embarcado selecionado foi o *Raspberry Pi 4 Model B,* junto com uma câmera

Raspberry Pi NoIR V2. A rede neural convolucional criada deveria ter uma acurácia em relação ao conjunto de validação superior a 95%.

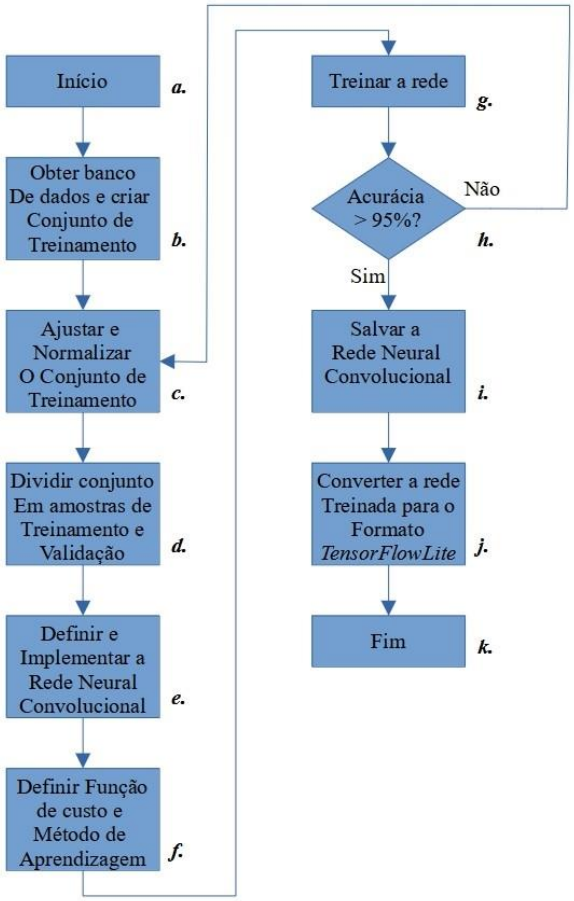
Após definir o hardware a ser utilizado, o desenvolvimento do projeto ocorreu de acordo com as seguintes etapas: Criação e treinamento da rede neural convolucional em um computador; Criação do programa para a aquisição e classificação de imagens usando a rede treinada; Implementação do programa na *Raspberry Pi*. As etapas de desenvolvimento do projeto estão na figura 06.

Figura 6: Etapas de desenvolvimento do projeto

Fonte: Autores

Para criar e treinar a rede neural convolucional, foram estipuladas algumas etapas, sendo elas: obter o banco de dados de imagens para a criação do conjunto de treinamento; ajustar e normalizar o conjunto de treinamento para o formato necessário; dividir o conjunto de treinamento entre amostras de treinamento e validação; definir e implementar a rede neural convolucional; definir função de custo e método de aprendizagem; treinar e salvar a rede. O fluxograma contendo as etapas para a definição e treinamento da rede está na figura 07.

Figura 7: Fluxograma de treinamento da RNC



Fonte: Autores

O conjunto de treinamento da rede RNC foi formado utilizando imagens de pessoas usando máscara, obtidas no banco de dados ***MaskedFace-Net*** (CABANI *et al.*, 2020, 1), em conjunto com imagens de pessoas sem máscara, obtidas no banco de dados ***Dataset Flickr-Faces-HQ (FFHQ)*** (AILA, KARRAS e LAINE, 2019, 8) e em conjunto com imagens de autoria própria. Ao todo, o conjunto de treinamento foi formado por 1000 imagens de pessoas sem máscara e 950 imagens de pessoas com máscara, resultando em 1950 amostras de treinamento, randomicamente distribuídas.

Cada amostra do conjunto de treinamento possuía inicialmente a dimensão 1024 x 1024 *pixels* no formato RGB24, sendo então redimensionados para uma dimensão de 224 x 224 *pixels* ainda no formato RGB24. A dimensão (224 x 224 *pixels*) é a dimensão da matriz de entrada selecionada para o modelo de rede neural artificial proposto neste artigo. A escolha deste valor, deve-se ao fato de que esta é uma dimensão de entrada comum a diversos outros modelos de redes neurais artificiais convolucionais, sendo um valor convencionalmente implícito. Além

disso, esta é uma dimensão de entrada que permite, caso necessário, o uso de outros modelos de câmera, tendo em vista que esta dimensão equivale a uma resolução de captura de imagem que poderá ser obtida em um grande número de modelos de câmera de video disponíveis no mercado.

As amostras de treinamento foram representadas computacionalmente na forma de matrizes da ferramenta e biblioteca *NumPy*, onde as matrizes de entrada da rede RNC, têm dimensão (224x224x3), sendo normalizadas e armazenadas em variáveis do tipo *float32*. As respectivas respostas ou rótulos, em inglês *labels*, foram representados pelo método *one-hot-encoding*, sendo o rótulo (1, 0) equivalente a resposta *com máscara* e o rótulo (0, 1) equivalente a resposta *sem máscara*.

Para a concepção da rede neural convolucional, foi utilizada a linguagem de programação Python em conjunto com as bibliotecas *TensorFlow/Keras*. A rede foi criada e treinada em um computador pessoal, sendo processada em uma unidade de processamento gráfico *NVIDIA RTX 2060* de 6Gbytes com *CUDA* e *Deep Neural Network library (CuDNN)*. A rede RNC criada possui, nesta ordem: Uma camada de entrada; Uma camada de 16 filtros convolucionais de dimensão 3 x 3, *stride* ou deslocamento 1 x 1 e função de ativação *ReLu*; Uma camada de *Pooling* do tipo *MaxPooling*, dimensão 2 x 2; Uma camada de 32 filtros convolucionais de dimensão 3 x 3, *stride* ou deslocamento 1 x 1 e função de ativação *ReLu*; Uma camada de *Pooling* do tipo *MaxPooling*, dimensão 2 x 2; Uma camada de operação do tipo *flatten*; Uma camada densa, ou seja, uma rede PMC, contendo 16 neurônios com função de ativação *ReLu*; Uma camada densa de saída PMC, contendo 2 neurônios com função de ativação *Sigmoid*.

A função de custo para o treinamento da rede RNC foi a função já citada anteriormente, *Binary Cross-Entropy*. O método de aprendizagem da rede ocorreu por meio do algoritmo *Stochastic Gradient Descent* com *momentum* (QIAN, 145, 1999), ficando a atualização dos pesos na seguinte forma:

$$v_t = \gamma v_{t-1} + \eta \nabla_w J(w; x_t; y_t) \tag{9}$$

$$w_{t+1} = w_t - v_t \tag{10}$$

Onde, γ é o parâmetro de *momentum*, w_t é o vetor de pesos no instante da iteração *t*, w_{t+1} é vetor de pesos cujos valores são atualizados imediatamente após a iteração *t*, η é a taxa de aprendizagem, $\nabla_w J(w; x_t; y_t)$ é o gradiente do erro ou função de custo em relação ao vetor de pesos w, $J(w; x_t; y_t)$ é a função de custo *Binary Cross-Entropy* em função do vetor de pesos w, x_t é a entrada da rede RNC no instante da iteração *t*, y_t é a saída da rede RNC no instante da

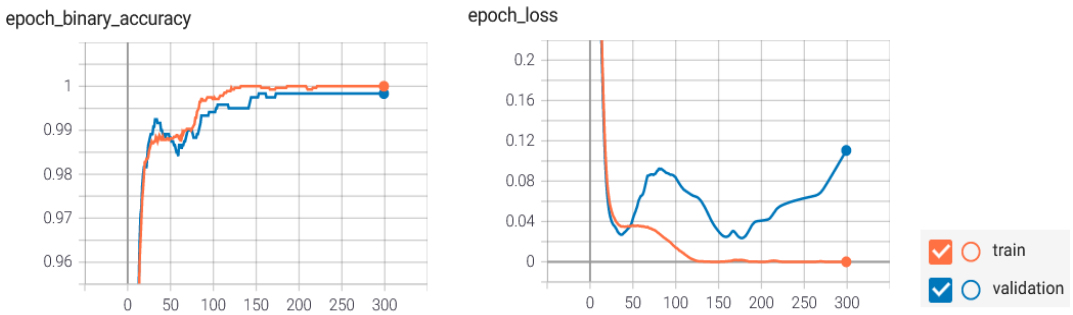
iteração *t*, v_t é uma expressão que contém e simplifica a compreensão do modo de funcionamento do SGD usando o termo *momentum*, v_{t-1} é equivalente a expressão v_t no instante da iteração anterior, $t - 1$.

A etapa de treinamento da rede foi realizada utilizando as ferramentas das bibliotecas *TensorFlow/Keras*, onde previamente foram especificados os parâmetros necessários a esta etapa, sendo eles: número de épocas, *batch size*, conjunto de treinamento e de validação, taxa de aprendizagem e *momentum.*

O número de épocas de treinamento especificado foi de 300 épocas, o valor para o *batch size* foi de 32, o valor da taxa de aprendizagem foi de $(.10)^{-6}$, o valor para o *momentum* foi de 0,9999, o valor de limiar para a acurácia da função de custo, *Binary Cross-Entropy*, foi de 0,5, o conjunto de treinamento foi dividido para formar o conjunto de validação, resultando em, 1365 amostras de treinamento e 585 amostras para validação.

Apesar de o valor do número de épocas ter sido especificado como 300 épocas, a rede convergiu para uma acurácia de 99% em relação ao conjunto de validação na época de número 83. Os gráficos da função de custo pelo número de épocas e acurácia em função do número de épocas de treinamento estão na figura 08.

Figura 8: Gráficos *TensorBoard* dos conjuntos de treinamento e validação pós treinamento. À esquerda, gráfico da (Acurácia x Época); À direita, gráfico do (Erro x Época)

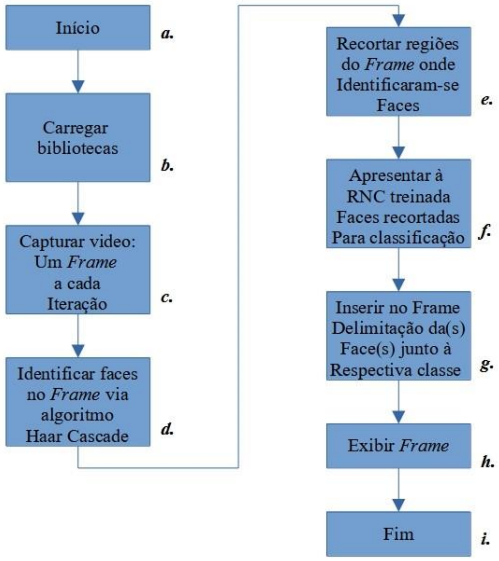


Fonte: Autores

Após a etapa de treinamento, pode-se observar a convergência da rede, para valores de acurácia em relação ao conjunto de validação e treinamento, acima do critério estipulado de 95%. Assim sendo, a rede foi salva em dois formatos, sendo eles o formato *Keras/TensorFlow* e *TensorFlowLite*.

Após a etapa de treinamento da rede RNC, foi necessário criar um programa na linguagem Python, para a aquisição de imagens por uma câmera, identificação de faces nestas imagens e a classificação das faces pela rede treinada. Este programa foi criado em computador convencional, fora do sistema embarcado *Raspberry Pi*.

Figura 9: Etapas do programa de aquisição e classificação de imagens usando a rede RNC treinada



Fonte: Autores

O programa de aquisição e classificação de imagens foi criado seguindo as etapas descritas na figura 09. O programa inicia com o carregamento das bibliotecas necessárias, sendo elas *OpenCV*, *NumPy*, *TensorFlow* e *OS*. Após o carregamento das bibliotecas, são carregadas a rede RNC treinada, o algoritmo de identificação de faces *Haar Cascade* e a seleção e ativação da câmera de video. Em seguida, são inseridos, em uma iteração do tipo *While* (Enquanto), os comandos para a captura da imagem pela câmera, na forma de *frame* (Quadro), sendo armazenado em uma variável do tipo *array* com uma representação RGB24 de dimensão 1920 x 1080. Imediatamente após este quadro ser obtido, ele é convertido em escalas de cinza e armazenado em uma nova variável, para ser utilizado no algoritmo *Haar Cascade*, onde ocorre a identificação de faces e suas regiões na imagem. As regiões onde foram identificadas as faces são recortadas, redimensionadas e normalizadas para serem enviadas à entrada da rede RNC, de modo a obter a classificação.

Como a rede possui dois neurônios de saída, a sua saída equivale a um vetor de duas variáveis, onde a combinação (1, 0) corresponde a classe **com máscara** e a combinação (0, 1) corresponde a classe **sem máscara**. No entanto a rede gera em sua saída, uma faixa de valores que vão de 0 a 1. Assim sendo, para que a interpretação da classe ocorresse da forma esperada, foi utilizado um valor de limiar de 0,5 entre as classes.

Ou seja, para o vetor de saída da rede RNC, os seus valores foram armazenados em uma variável do tipo *array* (vetor) denominada *classificacao*, variável esta que possui duas componentes, *classificacao[0][0]* correspondente a primeira componente, ou saída do primeiro neurônio da camada de saída e *classificacao[0][1]* correspondente a segunda componente, ou saída do segundo neurônio da camada de saída. Dado cada componente, foi feita a seguinte comparação para a interpretação da classe:

Se (classificacao[0][0] > 0,5) E (classificacao[0][1] < 0,5), então a classe é Com Máscara;

Senão Se (classificacao[0][0] < 0,5) E (classificacao[0][1] > 0,5), então a classe é Sem Máscara.

Para os casos em que a imagem de entrada não contenha faces, a rede classificará esta imagem como sendo pertencente à classe Sem Máscara. No entanto, esta situação é evitada pelo sistema, já que há uma pré-seleção de imagens que contenham faces pelo algoritmo de detecção de faces *Haar Cascade*. Ainda durante a identificação da classe de cada face, é desenhado um retângulo em torno da face e inserido um texto referente à classe na variável que armazena o quadro. O retângulo na cor azul corresponde à classe **com máscara** e o retângulo na cor vermelha corresponde à classe **sem máscara**. Já o texto inserido, por sua vez, é o nome da classe. Após a classificação ser realizada, o quadro é exibido na tela até que se pressione a tecla ‘p’.

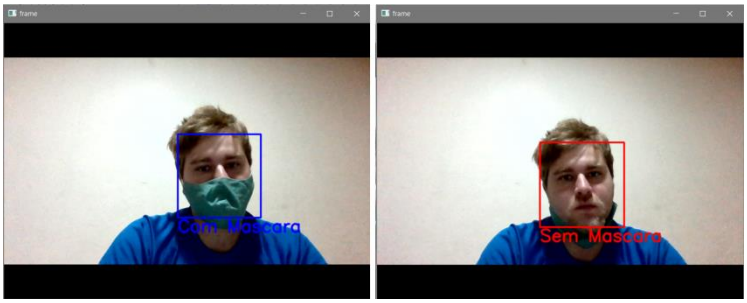
Para a implementação do código (programa) no sistema embarcado *Raspberry Pi*, foi necessário instalar o sistema operacional *Raspberry Pi OS* de 32-Bit, baseado no sistema operacional *Debian/Linux*, e em seguida as bibliotecas de programação para a linguagem Python, sendo elas, *OpenCV,* NumPy e *TensorFlow Lite.*

Devido à portabilidade da linguagem de programação Python, não houve alterações significativas no código. Algumas configurações para a captura de imagens pela câmera *Raspberry Pi NoIR V2* foram necessárias, já que esta câmera não possui o filtro para comprimentos de onda correspondente ao infravermelho e consequentemente a coloração das imagens capturadas não corresponde exatamente às cores reais.

Resultados

Um exemplo de resultado final da execução do programa pode ser observado na figura 10, onde são apresentados dois quadros: à esquerda, uma pessoa usando uma máscara e à direita a mesma pessoa sem máscara.

Figura 10: Resultado da classificação realizada pelo sistema. À esquerda, com máscara e à direita sem máscara.



Fonte: Autores

A acurácia do modelo treinado foi de 99% em relação ao conjunto de validação. No entanto, também foram apresentados 200 novos exemplos ao modelo, sendo 100 exemplos pertencentes à classe **Com máscara** e 100 exemplos pertencentes à classe **Sem máscara**. Foi gerada uma matriz de confusão (Tabela 1) comparando a saída predita pelo modelo e a saída esperada, resultando em 98 casos verdadeiro positivo (VP), nenhum caso falso negativo (FN), 2 casos falso positivo (FP) e 100 casos verdadeiro negativo (VN). Por caso positivo compreende-se **Com máscara** e por caso negativo compreende-se **Sem máscara**. Ou seja, para os 200 novos exemplos, apenas 2 exemplos foram classificados erroneamente.

Tabela 1 – Matriz de confusão

Esperada\ Predita	Com máscara	Sem máscara
Com máscara	98 (VP)	2 (FP)
Sem máscara	0 (FN)	100 (VN)

Durante a operação do sistema, alguns fatores externos, bem como iluminação e a cor da máscara, interferiram na identificação de faces pelo algoritmo *Haar Cascade*. No entanto, em um ambiente de iluminação controlada e com máscaras de cores azul, branca ou preta, o sistema não apresentou falhas. Ainda durante a operação do sistema, foi constatada uma taxa de quadros processados pouco menor que 5 quadros por segundo.

Conclusão

Neste projeto, foi proposto o desenvolvimento de um sistema embarcado que, com as imagens capturadas por uma câmera, é capaz de identificar e classificar faces quanto ao uso de máscara. Foram empregados diversos algoritmos relacionados à área do conhecimento da visão computacional, sendo o principal deles uma rede neural convolucional, que foi inicialmente treinada em um computador à parte, para então ser empregada no sistema embarcado *Raspberry Pi*.

A rede neural convolucional teve, durante a etapa de treinamento, uma convergência relativamente rápida e uma acurácia, em relação ao conjunto de validação, de 99%. O sistema proposto foi capaz de classificar faces quanto ao uso de máscara. No entanto, cabe ressaltar que, devido a rapidez de processamento do processador *ARM Cortex-A72* e do custo computacional relacionado aos algoritmos empregados, bem como a resolução da câmera, a taxa de exibição das imagens durante a classificação foi inferior a 5 quadros por segundo. Outro ponto importante a ressaltar é que, a depender das condições de iluminação do local, a identificação de faces e a classificação podem ser comprometidas, o que implica que este sistema deve ser empregado em condições de iluminação controladas e/ou adequadas.

Ainda assim, dentro de um ambiente com boa iluminação, o sistema demonstra-se eficaz para ser empregado como uma ferramenta auxiliar no controle e identificação de pessoas que estejam com ou sem máscara, podendo ser empregado durante pandemias análogas à do COVID-19, auxiliando no controle da disseminação da doença.

Referências

AILA, T.; KARRAS, T.; LAINE, S. A Style-Based Generator Architecture for Generative Adversarial Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. p. 4401- 4410, 2019.

BOTTOU, L. Stochastic Gradient Descent Tricks – Microsoft Research. In Neural networks: Tricks of the trade. Springer, Berlin, Heidelberg. p. 421-436, 2012.

CABANI, A. et al. MaskedFace-Net - A Dataset of Correctly/Incorrectly Masked Face Images in the Context of COVID-19**.** Smart Health 19. p.100144, 2020. Disponível em: 10.1016/j.smhl.2020.100144. Acesso em 23 Mai 2021.

CHEN, Tianqi et al. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853, 2015. Disponível em: https://arxiv.org/pdf/1505.00853.pdf. Acesso em 05 Mai. 2021.

CHERVONENKIS, A.Y.; VAPNIK, V. N. On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability and its Applications, v**.** 16, n. 2, p.264-80, 1971.

CHOWDARY, G. et al. Face mask detection using transfer learning of inceptionv3. In International Conference on Big Data Analytics. Springer, Cham, p. 81-90, 2020.

FARABET, C.; KAVUKCUOGLU, K.; LECUN, Y. Convolutional networks and applications in vision. In Proceedings of 2010 IEEE international symposium on circuits and systems. IEEE, p. 253-256, 2010.

GARRISON, C.; KANAN, C. Color-to-grayscale: Does the method matter in image recognition?. PloS one, v. 7, no. 1, 2012

GONZALEZ, R. C.; WOODS, R. E. Digital Image Processing, Editora Prentice Hall, 2º edição, 1992.

HE, K. et al. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. p. 770-778, 2016.

HO, Y.; WOOKEY, S. The Real World Cross-Entropy Loss Function: Modeling the Costs of Mislabeling – IEEE Access, v. 8, p. 4806-4813, 2019.

HOCHREITER, S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. World Scientific Publishing Company, v. 6, n. 02, p. 107-116, 1998.

JONES, M; VIOLA, P. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, v. 1, p. I-I, 2001.

LECUN, Y et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, v. 86, n. 11, p. 2278-2324, 1998.

LECUN, Yann et al. A theoretical analysis of feature pooling in visual recognition. In International Conference on Machine Learning (ICML), n. 27, p. 111–118, 2010.

LOEY, M. et al. Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. Sustainable Cities and Society, v. 65, p. 102600, 2021.

LOEY, M. et al. A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. Measurement, v. 167, p. 108288, 2021.

PLATANIOTIS, K.N.; VENETSANOPOULOS, A.N. Color Image Processing and Applications Engineering – Monograph (English), Editora Springer, Berlim, 2000.

QIAN, N, On the momentum term in gradient descent learning algorithms. Neural Networks: The Official Journal of the International Neural Network Society. v.12, n. 1, p.145-141, 1999.

SAÚDE, Organização Mundial de (OMS). Coronavirus disease (COVID-19): Masks, 2020. Disponível em: https://www.who.int/news-room/q-a-detail/coronavirus-disease-covid-19-masks#:~:text=Masks%20are%20a%20key%20measure,coughs%2C%20and%20more. Acesso em: 19 mar. 2021.

SAÚDE, Organização Mundial de (OMS). Mask use in the context of COVID-19 - Interim Guidance, 2020. Disponível em: https://apps.who.int/iris/bitstream/handle/10665/337199/WHO-2019-nCov-IPC_Masks-2020.5-eng.pdf?sequence=1&isAllowed=y. Acesso em 19 mar. 2021.

SAÚDE, Organização Mundial de (OMS). WHO Coronavirus (COVID-19) Dashboard, 2021. Disponível em: https://covid19.who.int/. Acesso em: 21 mar. 2021.

SILVA, I.; SPATTI, D.; FLAUZINO, R. Redes neurais artificiais para engenharia e ciências aplicadas. São Paulo: Editora Artliber, 2016.

SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. Proceedings of the IEEE conference on computer vision and pattern recognition. p. 2818-2826, 2016.

CLASSIFICATION OF IMAGES OF PEOPLE IN RELATION TO THE USE OF MASKS BY A CONVOLUTIONAL NEURAL NETWORK IMPLEMENTED IN AN EMBEDDED SYSTEM

ABSTRACT

This article discusses the development of an embedded system with a video camera. Based on the image obtained from it, aims to classify the images of individuals in two groups, "With a mask" or "Without a mask". Artificial intelligence algorithms were used to classify the individual's image, the main algorithm being based on Convolutional Artificial Neural Networks. The convolutional neural network created had an accuracy of 99% for the validation set and classified only 2 cases as false positive for a new set of 200 new examples. In operation, this system exhibits a frame rate of around 5 frames/s. For the identification of the faces to be classified the system requires that the environment has adequate lighting. This system is intended to be used during pandemics such as COVID-19 as a tool to control the spread of this virus.

Keywords: *Covid-19; Neural Networks; Mask; Pandemic; Image Classification.*

Envio: maio/2021
Aceito para publicação: junho/2021