# Modelo de Detecção do Uso de Máscaras Faciais Utilizando *Deep Learning*

**Ricardo Lucio Braga Reis - A57590919**
Análise Econômica e Geração de Valor - Professor Bernardo Aflalo
MBA in Business Analytics and Big Data (FGV) - Turma Berrini
ricardo.l.b.reis@gmail.com

## 1 Introdução

Desde o ínicio, a humanidade se depara com enfermidades capazes de dizimar populações, e com crescimento exponencial da população mundial e evolução da mobilidade, esse potencial de alastramento de doenças atingiu outro patamar recentemente com a disseminação da COVID-19. Somado à isso vem a falta de uma vacina que possa imunizar a população em tempo ágil e assim conter a doença, portanto lança-se mão de medidas preventivas capazes de diminuir a taxa de contágio.

De acordo com a OMS, a COVID-19 é transmitida pelo contato direto, indireto (através de superfícies e objetos contaminados) ou contato próximo com pessoas infectadas através de secreções da boca e nariz [5], portanto protocolos sociais foram desenvolvidos para diminuir a disseminação do vírus, dentre eles o uso de máscaras obrigatório em espaços públicos e estabelecimentos. Com isso surge a questão de como monitorar o uso de máscaras faciais na população de forma escalável como auxílio da tecnologia.

Segundo descrito por Naudé [4], tecnologias que utilizam IA (inteligência artificial) já vem sendo usadas mundo afora para fazer controle social, por exemplo em *Oxford* no Reino Unido uma câmera escaneia o público para detectar multidões, e na China, câmeras instaladas em aeroportos e estações de trem detectam a temperatura e o uso de máscaras faciais dos transeuntes.

Este artigo tem como objetivo geral implementar um modelo de detecção de pessoas utilizando máscaras faciais para posterior implementação em sistemas de controle de entrada em espaços públicos, estabelecimentos que executem atividades essenciais, repartições públicas estaduais, transporte por aplicativo para um público alvo de consumidores, fornecedores, clientes, empregados, colaboradores, agentes públicos e prestadores de serviço, utilizando técnicas de *deep learning*.

O artigo está organizado como segue: a Seção 2 apresenta o método utilizado no desenvolvimento do trabalho, a Seção 3 introduz os conceitos por trás dos algoritmos de inteligência artificial utilizados, em seguida a Seção 4 aborda o experimento desenvolvido, comparações e resultados, por fim, a Seção 5 discute as limitações e trabalhos futuros.

## 2 Metodologia

O ponto de partida deste estudo está na obtenção dos dados, portanto procura-se definir o banco de imagens de pessoas utilizando máscaras faciais a ser utilizado. Em seguida, através de um revisão bibliográfica busca-se as técnicas mais relevantes para o tema em questão a fim de estabelecer uma sequência de desenvolvimento. Posteriormente, com o treinamento dos modelos, verifica-se se o conjunto de imagens de treino, validação e teste estão adequados e se os resultados obtidos são relevantes para em seguida avaliar a performance e possível *tunning* de hiperparâmetros. Finalmente, avaliam-se os trabalhos futuros.

## 3 Referencial Teórico

### 3.1 ANN - *Artificial Neural Network*

Neste trabalho, optou-se por utilizar duas das técnicas de *deep learning* mais comuns para este tipo de aplicação, ANN (acrônimo de *Artificial Neural Network*) e CNN (acrônimo de *Convolutional Neural Network*). A natureza vem inspirando milhares de invenções, e não foi diferente com a inteligência artificial. De acordo com Géron [3], a ideia da primeira ANN surgiu em 1943 a partir de como neurônios biológicos deveriam funcionar em conjunto em um cérebro animal para performar tarefas.

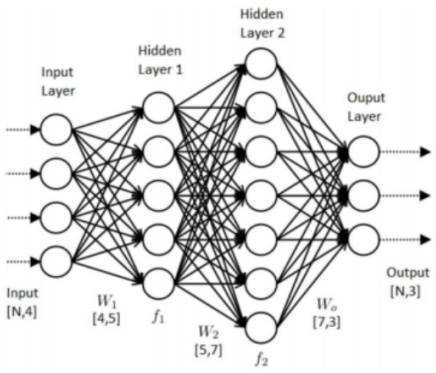



Figura 1: Arquitetura de uma rede neural.

É possível observar a representação de uma rede neural artificial na Figura 1, onde cada neurônio se interliga com os neurônios da próxima camada for-

mando assim uma rede conectada de forma densa (*dense layer*), e quanto mais camadas, mais complexa e profunda é essa rede, daí derivando a nomenclatura *deep learning*. Na Figura 2 observa-se como se relacionam os objetos em uma rede neural, logo, uma entrada de dados na rede sofre transformações camada a camada devido ao peso de cada neurônio. Ao final, obtém-se uma previsão (y') que é confrontada com a resposta verdadeira (y) através de uma função perda (*loss function*). O *optimizer* tem o papel de regular os pesos no neurônio a fim de minimizar o *loss score*, ou seja, melhorar a previsão. Assim se dá o processo de aprendizagem da rede a cada iteração.

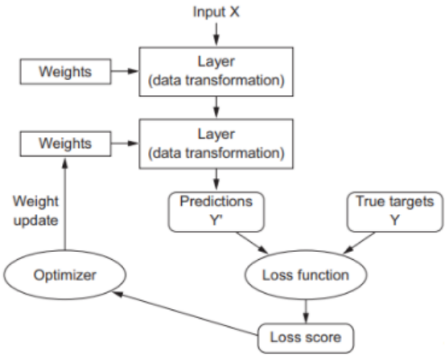



Figura 2: Ciclo de aprendizagem da rede.

Neste trabalho foi usado o otimizador SDG (acrônimo de *Stochastic Gradient Descent*), onde segundo Géron [3], a cada passo o algoritmo escolhe uma instância aleatoriamente do conjunto de dados para o cálculo do gradiente e ajuste dos pesos, ao invés de usar o conjunto todo ou partes pequenas (*batch* e *mini-batch*). Isso acarreta em maior velocidade no treinamento porém podendo nunca alcançar um mínimo global. Ao se treinar uma rede, eventualmente poderá ocorrer *overfitting*, ou seja, o modelo aprenderá com os dados de treino mas não conseguirá generalizar para novos dados. Para isso, utilizou-se uma técnica chamada *dropout* [2], onde a rede ajustará para zero o resultado de uma porcentagem de neurônios de cada camada, com o intuito de regularização.

### 3.2 CNN - *Convolutional Neural Networks*

As CNNs (acrônimo de *Convolutional Neural Networks*) são um tipo de modelo de *deep learning* voltado para visão computacional, e surgiram a partir do estudo do córtex visual onde observou-se que neurônios *high-level* são baseados na saída de neurônios *low-level* vizinhos, ou seja a partir da detecção de padrões mais simples combinados, um neurônio *high-level* detecta padrões complexos [3]. De forma análoga acontece no algoritmo através da camada de convolução, onde se aplicam filtros que destaquem determinados aspectos da imagem para posteriormente aplicar a função de ativação. Em seguida tem-se uma camada de *pooling* com o objetivo de diminuir a imagem vinda da camada convolucional a fim de reduzir a carga computacional, uso de memória, número de parâmetros e limitar o risco de *overfitting* [2][3].

Nas redes convolucionais, entre uma camada e outra utiliza-se a técnica de *batch normalization* para normalizar os dados ajudando na aprendizagem e generalização do modelo [2]. Ao final aplica-se uma *flatten*, que nada mais é que o achatamento de uma matriz (imagem) para um vetor, dado que ele entrará em uma camada densa, citada na seção anterior, para a classificação.

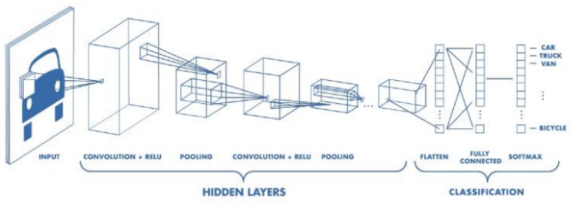



Figura 3: Arquitetura de uma rede neural convolucional.

## 4 Avaliação Experimental

### 4.1 Preparação

Como citado na Seção 2, o ponto de início deste trabalho foi a definição do banco de imagens de pessoas utilizando máscaras faciais, portanto utilizou-se um banco público [1][6] com 90.000 imagens de pessoas sem máscaras e 2.203 de pessoas com máscaras. Dado que a proporção de imagens com e sem máscaras estava desequilibrada, optou-se por fazer um balanceamento selecionando aleatoriamente 2.203 imagens de pessoas sem máscaras ficando assim com a proporção de 50%.

```
Tensor de features de treinamento:      (3524, 256, 256, 1)
Imagens:          3524
Sem máscara:      1762
Com máscara:      1762

Tensor de labels de treinamento:        (3524, 1)
Quantidade:       3524
Sem máscara:      1762
Com máscara:      1762

Tensor de features de teste:            (882, 256, 256, 1)
Imagens:          882
Sem máscara:      441
Com máscara:      441

Tensor de labels de teste:              (882, 1)
Quantidade:       882
Sem máscara:      441
Com máscara:      441
```

Figura 4: Quantidades nas bases de treino e teste.

Com o intuito de simplificar e diminuir o custo computacional optou-se também por fazer o redimensionamento das imagens para 256 x 256 pixels e conversão para escalas de cinza. Utilizou-se também a técnica de *data augmentation* para fazer o enriquecimento do banco de imagens aplicando pequenas alterações nas imagens como: rotação, translação, cisalhamento, ampliação, espelhamento a fim de gerar novas imagens para o treino do modelo. Por fim separou-se em conjunto de treinamento e de testes na proporção de 80%

para 20% e fez-se a normalização dos dados para a entrada nas redes.

Como citado na Seção 3, implementou-se duas arquiteturas de redes neurais. Observa-se na imagem abaixo a ANN formada por quatro camadas densas, sendo três com 512 neurônios e função de ativação *ReLu*, e a camada de saída *Sigmoid*, seguindo com um *Flatten* para a entrada de dados na rede e três *Dropouts* entre camadas de 40%, 30% e 10% respectivamente.

```
i = Input(x_train.shape[1:], name="entrada")
a = Flatten(name="achatamento")(i)
a = Dense(512, activation = 'relu', name="densa1")(a)
a = Dropout(0.4, name="reducao1")(a)
a = Dense(512, activation = 'relu', name="densa2")(a)
a = Dropout(0.3, name="reducao2")(a)
a = Dense(512, activation = 'relu', name="densa3")(a)
a = Dropout(0.1, name="reducao3")(a)
a = Dense(1, activation = 'sigmoid', name="previsao")(a)

model_NN = Model(i,a, name='NN')

model_NN.compile(optimizer = 'SGD',
                 loss = "binary_crossentropy",
                 metrics = ["accuracy"])
```

Figura 5: Arquitetura da ANN proposta.

Já a CNN foi construída com oito camadas convolucionais intercaladas por *max pooling* e *batch normalization* seguida de uma rede densa igual a descrita anteriormente. Para ambas as rede o otimizador escolhido foi o *Stochastic Gradient Descent*, *binary_crossentropy* como loss e acurácia como métrica.

```
i = Input(x_train.shape[1:], name="entrada")
b = Conv2D(32, (3,3), activation ='relu', padding = 'same', name="convolucao2d1")(i)
b = BatchNormalization(name="normalizacao1")(b)
b = Conv2D(32, (3,3), activation ='relu', padding = 'same', name="convolucao2d2")(b)
b = BatchNormalization(name="normalizacao2")(b)
b = MaxPooling2D(2,2, name="acumulacao1")(b)
b = Conv2D(64, (3,3), activation ='relu', padding = 'same', name="convolucao2d3")(b)
b = BatchNormalization(name="normalizacao3")(b)
b = Conv2D(64, (3,3), activation ='relu', padding = 'same', name="convolucao2d4")(b)
b = BatchNormalization(name="normalizacao4")(b)
b = MaxPooling2D(2,2, name="acumulacao2")(b)
b = Conv2D(128, (3,3), activation ='relu', padding = 'same', name="convolucao2d5")(b)
b = BatchNormalization(name="normalizacao5")(b)
b = Conv2D(128, (3,3), activation ='relu', padding = 'same', name="convolucao2d6")(b)
b = BatchNormalization(name="normalizacao6")(b)
b = MaxPooling2D(2,2, name="acumulacao3")(b)
b = Conv2D(256, (3,3), activation ='relu', padding = 'same', name="convolucao2d7")(b)
b = BatchNormalization(name="normalizacao7")(b)
b = Conv2D(256, (3,3), activation ='relu', padding = 'same', name="convolucao2d8")(b)
b = BatchNormalization(name="normalizacao8")(b)
b = MaxPooling2D(2,2, name="acumulacao4")(b)
b = Flatten(name="achatamento")(b)
b = Dense(512, activation = 'relu', name="densa1")(b)
b = Dropout(0.4, name="reducao1")(b)
b = Dense(512, activation = 'relu', name="densa2")(b)
b = Dropout(0.3, name="reducao2")(b)
b = Dense(512, activation = 'relu', name="densa3")(b)
b = Dropout(0.1, name="reducao3")(b)
b = Dense(1, activation = 'sigmoid', name="previsao")(b)

model_CNN = Model(i, b, name='CNN')

model_CNN.compile(optimizer = 'SGD',
                  loss = "binary_crossentropy",
                  metrics = ["accuracy"])
```

Figura 6: Arquitetura da CNN proposta.

## 4.2 Resultados

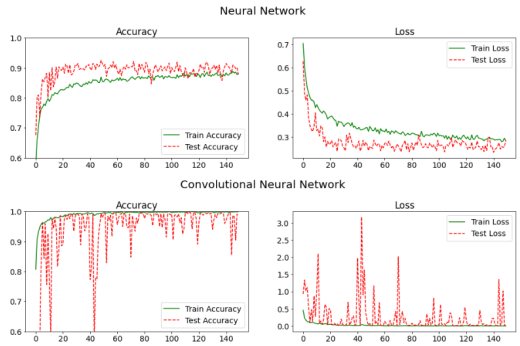
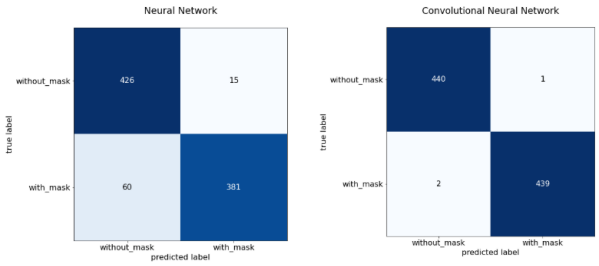



Figura 7: *Accuracy* e *loss* das redes ANN e CNN.

Utilizando 150 épocas de treino para as duas redes chegou-se ao seguinte resultado apontado nas Figuras 7, 8, 9 e 10. Apesar da ANN ter uma acurácia boa de 91% com 75 erros, a CNN chega a quase 100% de acerto, errando apenas 3 imagens, como é possível observar na matriz de confusão. É possível notar na Figura 7 que, apesar da CNN ter uma evolução mais desordenada, ela converge mais rápido, enquanto que na ANN acontece a partir da época 100. Já na curva ROC, a CNN obteve o valor 1 enquanto a ANN 0.98, o que também é um resultado satisfatório.




(a) ANN (b) CNN

Figura 8: Matriz de confusão das redes ANN e CNN.

Analisando o *precision* e *recall*, observa-se que na ANN, para a classificação de pessoas sem máscaras os indicadores atingiram 0.88 e 0.99, e de pessosas com máscaras atingiram 0.96 e 0.86 respectivamente. Já a CNN obteve 100% em todos os quesitos.

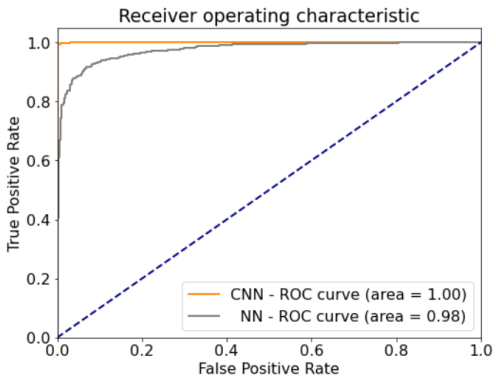



Figura 9: Curva ROC.

```
Classification Report - Neural Network
              precision    recall  f1-score   support

           0       0.88      0.97      0.92       441
           1       0.96      0.86      0.91       441

    accuracy                           0.91       882
   macro avg       0.92      0.91      0.91       882
weighted avg       0.92      0.91      0.91       882


Classification Report - Convolutional Neural Network
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       441
           1       1.00      1.00      1.00       441

    accuracy                           1.00       882
   macro avg       1.00      1.00      1.00       882
weighted avg       1.00      1.00      1.00       882
```

Figura 10: Medidas de *accuracy*, *precision*, *recall* e *F1-score* dos modelos.

# 5 Conclusão

Este trabalho discutiu e implementou um modelo de detecção de máscaras faciais utilizando *deep learning* visando assim cumprir uma demanda por controle social no combate ao alastramento da COVID-19. Entende-se que existe um trabalho ainda a ser feito por especialistas em direito e ciências sociais que lidem com aspectos legais, éticos e regulatórios do uso de inteligência artificial.

Este trabalho limitou-se ao estudo e implementação do modelo, portanto o mesmo não foi testado em ambiente de produção ou integrado à *hardware* de *computer vision*. Como trabalhos futuros, seguindo nessa linha de controle social no combate ao alastramento do Coronavírus, novas implementações podem contemplar o desenvolvimento de modelos que detectem se clientes estão mantendo o distanciamento de 1,5m por exemplo, densidade de pessoas que caracterizem aglomerações, sentido ao caminhar, com o intuito de auxiliar lojas na implantação de corredores de mão única, modelos de aferição de temperatura corporal e a responsabilidade social na utilização de *machine learning* para o bem estar de todos.

# Referências


[1] Zhongyuan Wang et al. "Masked Face Recognition Dataset and Application". Em: (2020).

[2] François Chollet. *Deep Learning with Python*. Manning, 2018. ISBN: 9781617294433.

[3] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn TensorFlow: Concepts, Tools, And Techniques To Build Intelligent Systems*. O'Reilly, 2017. ISBN: 978-1-491-96229-9.

[4] Wim Naudé. "Artificial Intelligence against COVID-19: An Early Review." Em: (2020). ISSN: 2365-9793.

[5] World Health Organization. *QA: How is COVID-19 transmitted?* URL: `https://www.who.int/news-room/q-a-detail/q-a-how-is-covid-19-transmitted`. (acessado em: 14.07.2020).

[6] Zhangyang Xiong. *Real-World Masked Face Dataset*. URL: `https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset`.