Nome Completo: Rodrigo Jose Borges Goncalves / Matrícula: A57566093

Mini Paper Individual

Instruções

O trabalho poderá ser feito em até 5 pessoas.

O objetivo do trabalho deve contemplar assuntos da aula (deep neural network, convolutional neural network, recurrent neural network) e deve ser escolhido por cada grupo. Na nossa última aula, dia 15/06, os alunos devem apresentar o grupo e qual o tema e o objetivo específico do trabalho. Ex.: 'O grupo, composto pelos alunos ....., terá como trabalho a estimativa dos casos de covid no Brasil utilizando arquiteturas de redes recorrentes, como LSTM e GRU.' Os grupos que tiverem dificuldade de escolher projetos, podem entrar em contato comigo. Algumas sugestões/ inspirações podem ser obtidas em sites de competição (ex Kaggle), mas a solução final deve conter elementos originais.

A entrega final deve ser feita individualmente por cada aluno até o dia 31/07 e deverá conter:

Os notebooks/ scripts com o trabalho feito e as soluções produzidas pelo grupo (comum a todos os membros do grupo);

Documentação individual, explicando o que foi feito no trabalho, problemas encontrados e possibilidades de melhoria da solução, em formato de 'mini papers' (2-5 páginas). Cada um deve ter a sua submissão de mini-paper.

31/julho: Entrega do notebook / scripts com o trabalho e o mini paper individual, no EClass.

Sumário

Sumário 2
Introdução 3
Contexto 3
Notebook do Grupo 4
Detalhamento do Desenvolvimento 4
Conclusão 11
Referências 12

Introdução

Este documento se refere a submissão do mini paper individual solicitado como parte da avaliação da disciplina Análise Econômica e Geração de Valor.

O projeto desenvolvido pelo grupo foi uma extensão do tema explorado na disciplina Desafios e Requisitos dos Projetos Analíticos.

Contexto

No contexto atual de propagação da pandemia do COVID-19 optamos, como grupo por desenvolver um trabalho alinhado ao contexto da nova realidade imposta em nossa sociedade em relação a medidas de controle social, mais especificamente em relação a recomendação e determinação de uso de máscaras faciais em locais de acesso público.

No início de maio de 2020 o estado de São Paulo decretou o uso de máscaras de proteção facial como obrigatório, por tempo indeterminado. A medida, que entrou em vigor em 07/05/2020, estabelece o uso de máscaras em: espaços públicos, estabelecimentos que executem atividades essenciais, repartições públicas estaduais e transporte por aplicativo.

Recentemente, em 3 de Julho de 2020 também foi sancionado pelo presidente da república, com alguns vetos, a Lei nº 14.019 de 02/07/2020 que determina o uso obrigatório de máscaras de proteção individual para circulação em espaços públicos e privados acessíveis ao público, em vias públicas e em transportes públicos durante a vigência das medidas para enfrentamento da emergência de saúde pública de importância internacional decorrente da pandemia da Covid-19.

Sob este contexto decidimos aplicar algumas técnicas de *deep learning* para o desenvolvimento de um modelo capaz de detectar a utilização de máscara de proteção individual.

Notebook do Grupo

Este trabalho está disponível em um notebook *Jupyter*, escrito em *Pyhton* 3, desenvolvido na plataforma *kaggle* que pode ser acessado no endereço a seguir: https://www.kaggle.com/leandrodaniel/fgv-mba-analise-economica-e-geracao-de-valor

Este notebook também pode ser executado a partir utilizando o *Binder*, pelo *link*: https://mybinder.org/v2/gh/ldaniel/Artificial-Intelligence-Applications/master

Detalhamento do Desenvolvimento

Para este trabalho utilizamos uma base de dados pública contendo um total de 90.000 imagens de faces sem máscaras e 2.203 imagens de faces com máscaras faciais.

Porém por alguma limitação o *kaggle* não reconheceu as 90.0000 imagens de faces sem máscaras que estão em nosso repositório no *GitHub* e limitou a quantidade de imagens sem máscaras em 56.564 imagens.

```
Dataset original

 58767 imagens no dataset com o tamanho (256, 256)

Proporção
Sem máscara:     56564
Com máscara:     2203
```

Porém esta limitação do *kaggle* não foi um problema pois executamos um balanceamento de nosso *dataset* onde selecionamos aleatoriamente 2.203 imagens das 56.564 para termos um *dataset* com 50% de imagens com máscaras e 50% de imagens sem máscaras.

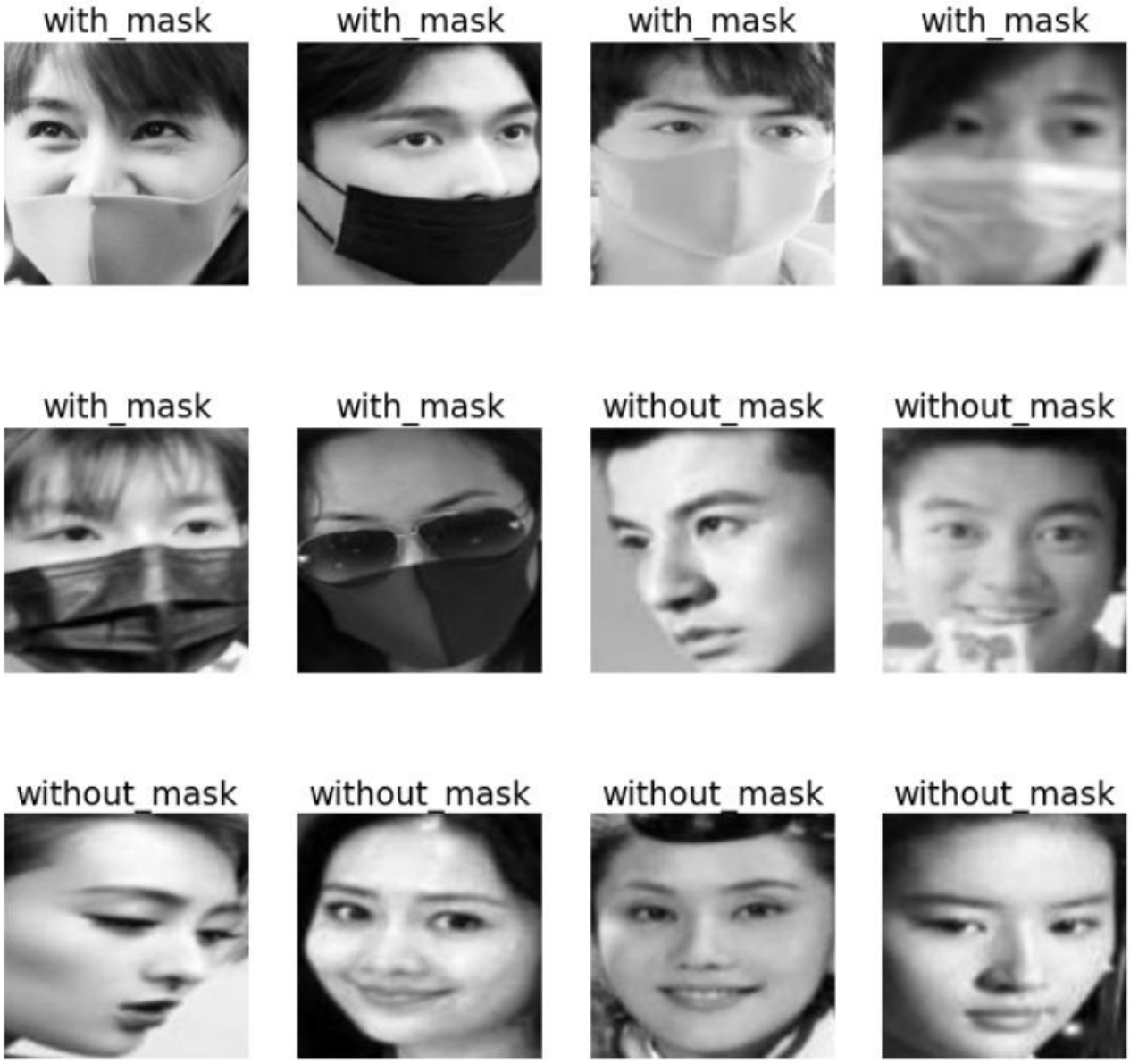
Executamos este passo pois vamos utilizar a métrica de acurácia como benchmark de análise da performance dos modelos e se não os fizéssemos um modelo que classificasse todas as amostras como não utilizando máscaras faciais já teria uma acurácia de 97.61%.

```
Dataset rebalanceado (com seleção aleatória da classe majoritária)

 4406 imagens no dataset com o tamanho (256, 256)

Proporção
Sem máscara:     2203
Com máscara:     2203
```

Para a simplificação do modelo também optamos por transformar as imagens para *GREYSCALE* utilizando as funções do *framework* de *computer vision* para *python* opencv-

python. Também executamos o redimensionamento das imagens para as dimensões 256p por 256p.

Este processamento nos retorna um tensor contendo uma matriz por imagem com 256 por 256 *pixels* com valores variando de 0 a 255 representando as diversas gradações de cinza entre branco e preto. Abaixo podemos visualizar algumas imagens após as transformações mencionadas acima:



Na sequência executamos a normalização dos valores das nossas matrizes dividindo o valor de cara elemento da matriz por 255, de forma que os valores se encontrem entre 0 e 1 para facilitar o processo de treinamento do modelo.

Realizamos o split do *dataset*, utilizando a biblioteca scikit-learn, entre base de treino e base de testes na proporção de 80% para treinamento e 20% para teste e validação. E criamos dois tensores para cada *dataset*, um contendo as matrizes representativas dos *pixels* das imagens e um outro vetor representando o *label* de cada imagem.

```
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
```

Na sequência do trabalho desenhamos duas rendes neurais. A primeira rede trata-se de uma rede neural simples contendo uma camada *flatten* utilizada para fazer a transformação da matriz representativa da imagem em um vetor, seguida de apenas 3 camadas densas (*hiden layers*) de 512 neurônios cada, utilizando a função de ativação *ReLU* e com um *dropout* de 40%, 30% e 10% entre cada *hiden layers*, seguida de uma camada densa de apenas um neurônio com a função de ativação *sigmoid* que retorna o equivalente a probabilidade de a observação em questão ser uma imagem de uma face utilizando máscara de proteção facial.

Como otimizador utilizados *Stochastic Gradient Descent* onde o cálculo dos ajustes dos pesos e bias da matriz de pesos é feito com base na derivada parcial de cada *batch* de imagens (utilizamos um *batch size* de 32 imagens), a função de perda utilizada foi a *Binary Crossentropy* que é a recomendada para problemas de solução binária como o nosso caso, e por fim estamos utilizando a métrica acurácia para a seleção da do melhor modelo.

Esta rede neural tem um total de 34.080.769 de parâmetros treináveis.

```
Model: "NN"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
entrada (InputLayer)         [(None, 256, 256, 1)]     0
_________________________________________________________________
achatamento (Flatten)        (None, 65536)             0
_________________________________________________________________
densa1 (Dense)               (None, 512)               33554944
_________________________________________________________________
reducao1 (Dropout)           (None, 512)               0
_________________________________________________________________
densa2 (Dense)               (None, 512)               262656
_________________________________________________________________
reducao2 (Dropout)           (None, 512)               0
_________________________________________________________________
```

```
densa3 (Dense)               (None, 512)               262656
_________________________________________________________________
reducao3 (Dropout)           (None, 512)               0
_________________________________________________________________
previsao (Dense)             (None, 1)                 513

=================================================================
Total params: 34,080,769
Trainable params: 34,080,769
Non-trainable params: 0
=================================================================
```

Na sequência da rede neural simples especificamos uma etapa convolucional com um total de 8 camadas convolucionais, utilizando a função de ativação *ReLU* e com diferentes parâmetros de número de filtros e com o *padding* = *same* para a manutenção das dimensões das matrizes entre as camadas, também entre as camadas utilizamos o processo de *MaxPooling2D* para utilizado para obter uma representação da uma imagem com a redução de sua dimensionalidade permitindo suposições sobre os recursos contidos nas sub-regiões classificadas.

A saída da etapa convolucional é acoplada a entrada da rede neural densa especificada inicialmente para a finalização do processo de predição. A etapa convolucional funciona como uma fase de *feature engineering* para processos de *computer vision* onde a rede aprende coo identificar as características importantes das imagens fazendo o *enhance* destas características que facilitam o processo de aprendizado. Nossa rede convolucional tem um total de 35,254,369 de parâmetros treináveis.

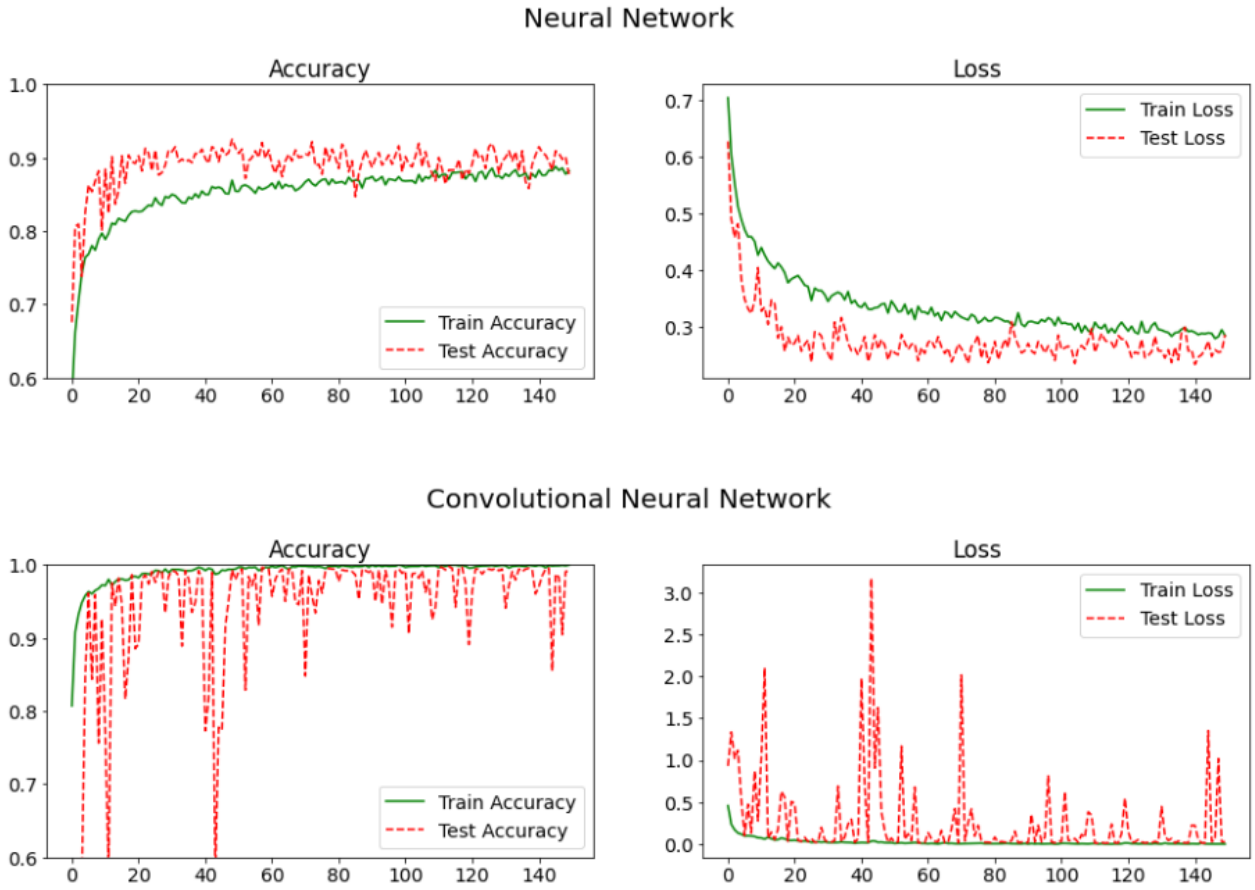
Esta arquitetura de rede vem sendo utilizada pelo nosso grupo com bastante sucesso em outros trabalhos como detecção de pneumonia em imagens de raio x, (acurácia de 96%) e classificação de classificação de dígitos mnist (acurácia de 99.56%).

```
Model: "CNN"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
entrada (InputLayer)         [(None, 256, 256, 1)]     0
_________________________________________________________________
convolucao2d1 (Conv2D)       (None, 256, 256, 32)      320
_________________________________________________________________
normalizacao1 (BatchNormaliz (None, 256, 256, 32)      128
_________________________________________________________________
convolucao2d2 (Conv2D)       (None, 256, 256, 32)      9248
_________________________________________________________________
normalizacao2 (BatchNormaliz (None, 256, 256, 32)      128
_________________________________________________________________
acumulacao1 (MaxPooling2D)   (None, 128, 128, 32)      0
_________________________________________________________________
convolucao2d3 (Conv2D)       (None, 128, 128, 64)      18496
_________________________________________________________________
normalizacao3 (BatchNormaliz (None, 128, 128, 64)      256
_________________________________________________________________
convolucao2d4 (Conv2D)       (None, 128, 128, 64)      36928
_________________________________________________________________
normalizacao4 (BatchNormaliz (None, 128, 128, 64)      256
```

```
_________________________________________________________________
acumulacao2 (MaxPooling2D)   (None, 64, 64, 64)        0
_________________________________________________________________
convolucao2d5 (Conv2D)       (None, 64, 64, 128)       73856
_________________________________________________________________
normalizacao5 (BatchNormaliz (None, 64, 64, 128)       512
_________________________________________________________________
convolucao2d6 (Conv2D)       (None, 64, 64, 128)       147584
_________________________________________________________________
normalizacao6 (BatchNormaliz (None, 64, 64, 128)       512
_________________________________________________________________
acumulacao3 (MaxPooling2D)   (None, 32, 32, 128)       0
_________________________________________________________________
convolucao2d7 (Conv2D)       (None, 32, 32, 256)       295168
_________________________________________________________________
normalizacao7 (BatchNormaliz (None, 32, 32, 256)       1024
_________________________________________________________________
convolucao2d8 (Conv2D)       (None, 32, 32, 256)       590080
_________________________________________________________________
normalizacao8 (BatchNormaliz (None, 32, 32, 256)       1024
_________________________________________________________________
acumulacao4 (MaxPooling2D)   (None, 16, 16, 256)       0
_________________________________________________________________
achatamento (Flatten)        (None, 65536)             0
_________________________________________________________________
densa1 (Dense)               (None, 512)               33554944
_________________________________________________________________
reducao1 (Dropout)           (None, 512)               0
_________________________________________________________________
densa2 (Dense)               (None, 512)               262656
_________________________________________________________________
reducao2 (Dropout)           (None, 512)               0
_________________________________________________________________
densa3 (Dense)               (None, 512)               262656
_________________________________________________________________
reducao3 (Dropout)           (None, 512)               0
_________________________________________________________________
previsao (Dense)             (None, 1)                 513

=================================================================
Total params: 35,256,289
Trainable params: 35,254,369
Non-trainable params: 1,920
=================================================================
```

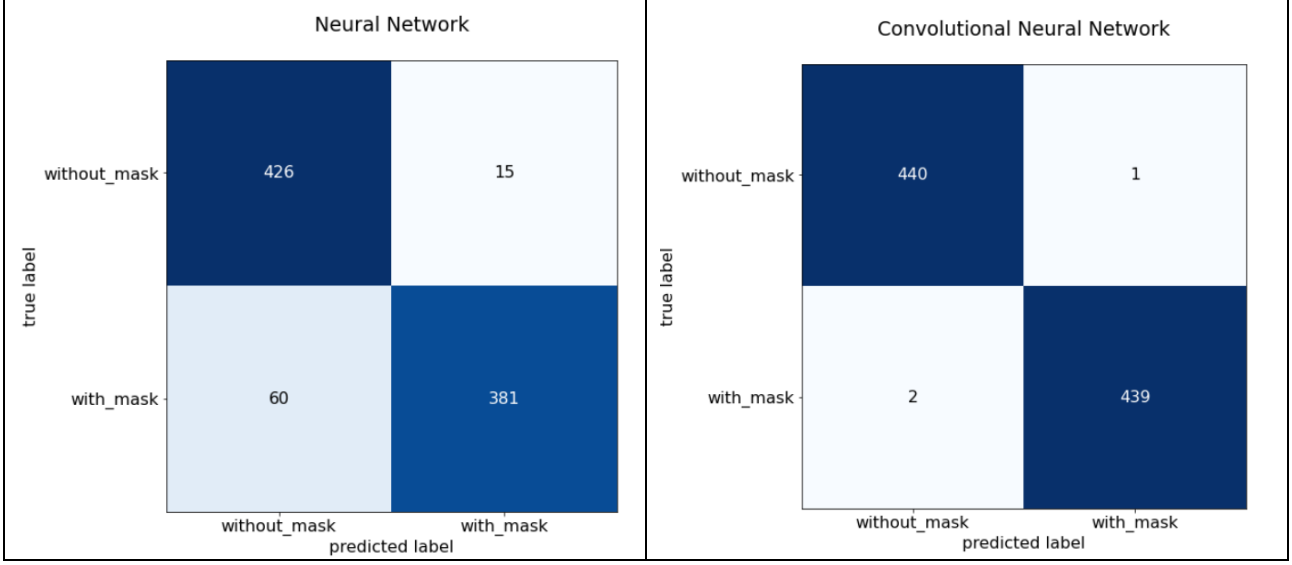
Também utilizamos algumas técnicas de data augmentation que consiste em aplicar algumas alterações nas imagens como rotação, *flip* horizontal, distorções, *rescale* e *zoom* nas imagens de forma a obter uma maior massa de dados para o processo de treinamento.

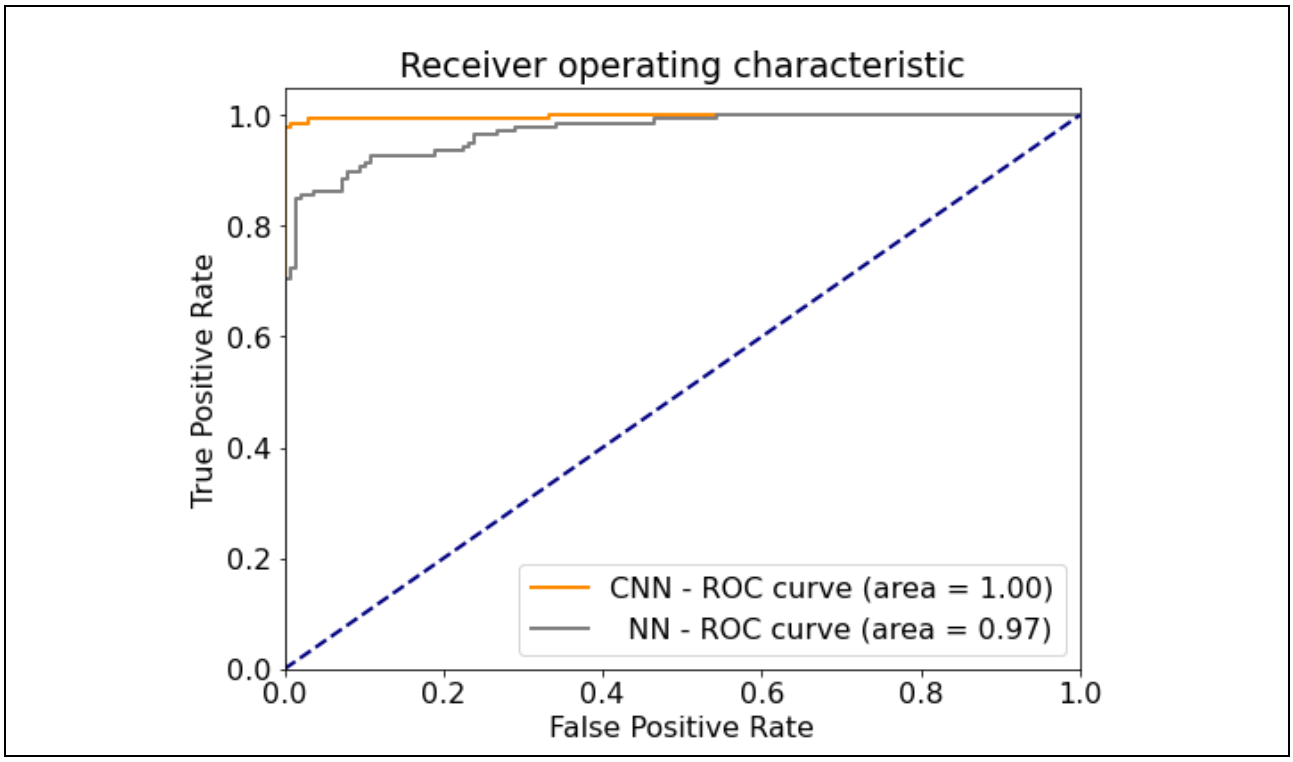
Treinamos dois modelos, um com a etapa convolucional e outro apenas com a etapa densa por 150 épocas, abaixo podemos visualizar o comportamento da curva de aprendizado de amos os modelos em função do resultado de nossa função de perda (*binary crossentroy*) e da acurácia dos mesmos nas bases de treino e testes, conseguimos verificar que o processo de aprendizado da rede CNN apesar de ser mais errático na base de treino acaba convergindo rapidamente na base de testes chegando a alcançar uma acurácia em testes de 99.66% frente a 91.5% da rede neural com apenas as camadas densas.



Nossa rede neural simples (sem as camadas convolucionais) 75 erros enquanto nossa rede convolucional alcançou a incrível marca de apenas 3 erros no mesmo *dataset*.

```
Neural Network                          Convolutional Neural Network
quantidade de erros:    75              quantidade de erros:    3
quantidade de hits:     807             quantidade de hits:     879

Acurácia:       0.9149659863945578      Acurácia:       0.9965986394557823
```





Um exercício final que realizamos é olhar para algumas imagens que foram classificadas incorretamente para verificarmos o quão razoável são os erros de cada modelo, vamos iniciar pelo modelo mais simples.

Como podemos observar no erros do modelo o mesmo não foi bem sucedido em algumas imagens que são muito fáceis de classificação por nos seres humanos, o que indica que o modelo com apenas as camadas densas não é capaz de capturar suficientemente os nuâncias das imagens a fins de fazer um previsão com alta qualidade.

Porém o modelo convolucional foi ainda mais surpreendente, os 2 dos míseros 3 erros do modelo na realidade são de pessoas utilizando a mascada de proteção de forma incorreta, estas imagens estavam na realidade mal classificadas em nosso *dataset*.

Conclusão

A resposta ao problema de negócio referente a detecção de do uso de máscara de proteção facial em imagens de faces foi plenamente atendida no desenvolvimento dos modelos selecionados, para a implementação do mesmos é necessário a aplicação de uma fase preliminar para o recorte das faces presentes em uma imagem que pode ser realizado utilizando um framework aberto como o *OpenCV*, uma vez que as imagens das faces são extraídas nosso modelo é capaz de classificar com acurácia aceitável a utilização ou não de máscara de proteção individual.

Referências

CHOLLET, F. **Deep Learning with Python**. Manning. ISBN 9781617294433. NY, USA. 2018.

Keras API Reference. Disponível em https://keras.io/api. Acesso em: 18 de Julho de 2020

Scikit Learn Documentation. Disponível em https://scikit-learn.org/stable/. Acesso em 18 de julho de 2020.

OpenCV Python Documentation. Disponível em https://pypi.org/project/opencv-python/. Aceso em 18 de julho de 2020.

CAMPOS, D; DANIEL L; REIS, R; GONCALVES, R; LIMA Y. **FGV-MBA - Detecção de Máscara Facial.** Disponível em https://github.com/ldaniel/Challenges-Requirements-Analytical-Projects.

CAMPOS, D; DANIEL L; REIS, R; GONCALVES, R; LIMA Y. **FGV-MBA - Análise Econômica e Geração de Valor**. Disponível em https://www.kaggle.com/leandrodaniel/fgv-mba-analise-economica-e-geracao-de-valor.

CAMPOS, D; DANIEL L; GONCALVES, R; LIMA Y. **FGV-MBA - Advanced-Predictive-Analysis-CNN-Implementation**. Disponível em https://www.kaggle.com/rodrigonca/advanced-predictive-analysis-cnn-implementation.

GONCALVES, R. **FGV-MBA - Análise Preditiva Avançada** - Trabalho Individual. Disponível em https://www.kaggle.com/rodrigonca/an-lise-preditiva-avan-ada-trabalho-individual.