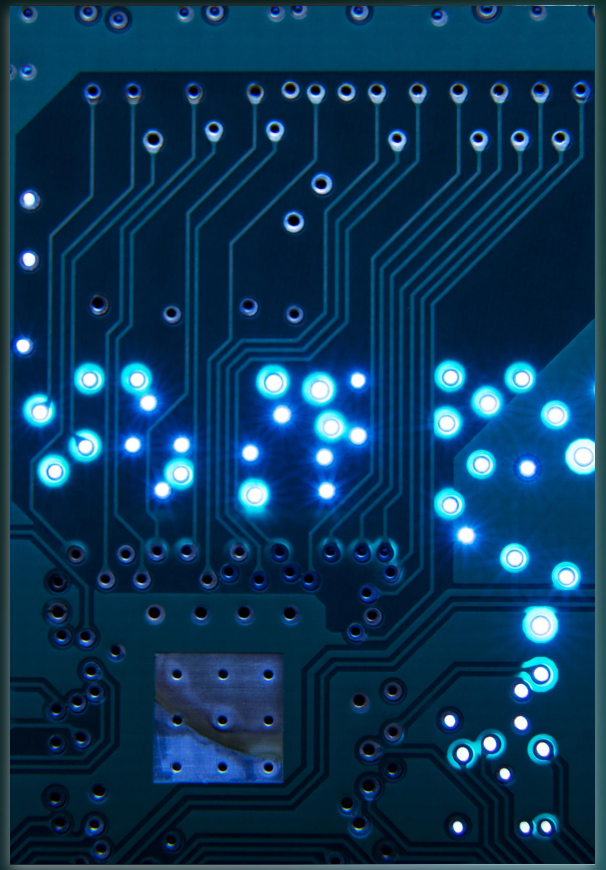
AVANTI FINGERS_

Equipe 6

SUMÁRIO

01

INTRODUÇÃO_



INTRODUÇÃO

RESUMO DO PROBLEMA

O problema em questão é a detecção de dedos em mãos utilizando técnicas de aprendizado de máquina.

PORQUE ESSE PROBLEMA?

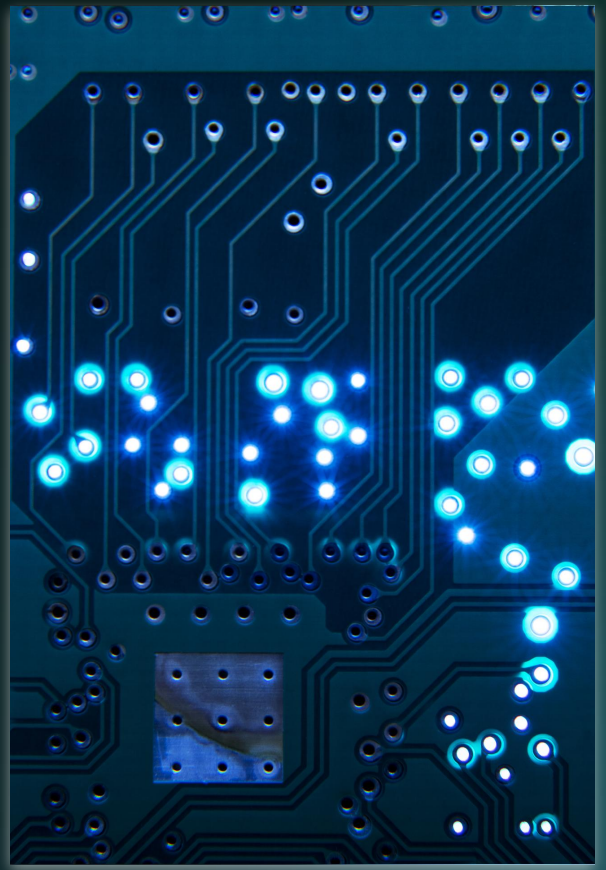
Em muitos contextos, como robótica, automação e análise de vídeo, compreender as ações das mãos humanas é crucial para inferir a intenção e o comportamento humano.

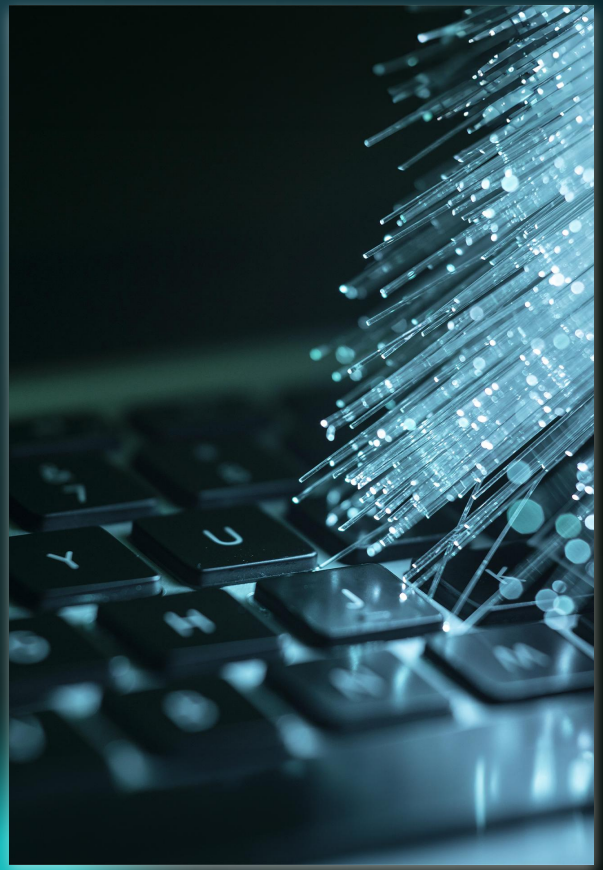
OBJETIVO GERAL

O objetivo do projeto é construir um modelo capaz de contar os dedos e de distinguir entre a mão esquerda e a mão direita.

02

METODOLOGIA_





BASE DE DADOS

As imagens foram retiradas de um dataset do site Kaggle:

1. As mãos são separadas em esquerda e direita, com 9000 imagens para cada.

2. São separadas 3000 imagens para cada quantidade de dedos(0, 1, 2, 3, 4 e 5), com 1500 para mão esquerda e 1500 para a direita

3. Totalizando 18000 imagens.

QUANTIDADE DE IMAGENS POR CLASSE

DEDOS: 0
LADO: L
QNT: 1500

DEDOS: 0
LADO: R
QNT: 1500

DEDOS: 1
LADO: L
QNT: 1500

DEDOS: 1
LADO: R
QNT: 1500

DEDOS: 2
LADO: L
QNT: 1500

DEDOS: 2
LADO: R
QNT: 1500

DEDOS: 3
LADO: L
QNT: 1500

DEDOS: 3
LADO: R
QNT: 1500

DEDOS: 4
LADO: L
QNT: 1500

DEDOS: 4
LADO: R
QNT: 1500

DEDOS: 5
LADO: L
QNT: 1500

DEDOS: 5
LADO: R
QNT: 1500

CÓDIGO ESTUDADO

Foi estudado os códigos dos seguintes notebooks :

- Count Finger Accurancy[100%](https://www.kaggle.com/code/muki2003/count-finger-accurancy-100/notebook)
- Transfer Learning 99% Accuracy Tensorflow (https://www.kaggle.com/code/ahmadmustafa12/transfer-learning-99-accuracy-tensorflow)

Com o mesmo modelo do código estudado (99%)

O modelo usado é uma combinação de um extrator de características pré-treinado, VGG16, seguido por um classificador customizado. Em outras palavras, é utilizada uma rede neural pré-treinada para extrair características das imagens, e depois adiciona e treina um classificador em cima dessas características.

Model: "vgg16"

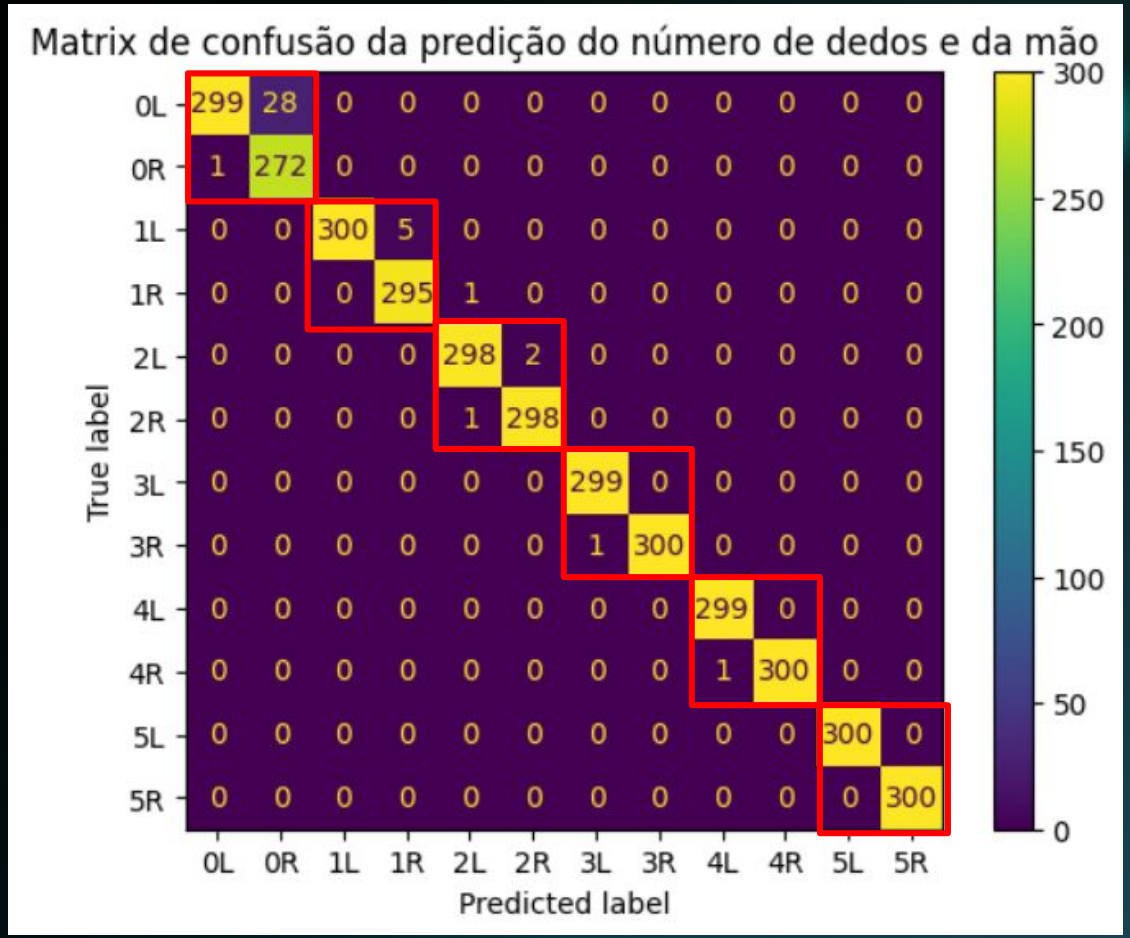
Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 128, 128, 3)	0
block1_conv1 (Conv2D)	(None, 128, 128, 64)	1,792
block1_conv2 (Conv2D)	(None, 128, 128, 64)	36,928
block1_pool (MaxPooling2D)	(None, 64, 64, 64)	0
block2_conv1 (Conv2D)	(None, 64, 64, 128)	73,856
block2_conv2 (Conv2D)	(None, 64, 64, 128)	147,584
block2_pool (MaxPooling2D)	(None, 32, 32, 128)	0
block3_conv1 (Conv2D)	(None, 32, 32, 256)	295,168
block3_conv2 (Conv2D)	(None, 32, 32, 256)	590,080
block3_conv3 (Conv2D)	(None, 32, 32, 256)	590,080
block3_pool (MaxPooling2D)	(None, 16, 16, 256)	0
block4_conv1 (Conv2D)	(None, 16, 16, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 16, 16, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 16, 16, 512)	2,359,808

Diferenças:

- Tamanho das imagens reduzido (de 224 para 128)

Resultados:

- Maior velocidade no processamento (treino e teste);
- Modelo bem mais leve;
- Accuracy mantida



Com CNN do Zero

Por conta dos notebooks estudados serem construídos em cima de modelos já treinados com imagens mais gerais, nos perguntamos qual seria a efetividade e as vantagens de criar e treinar uma CNN do zero, apenas com imagens do dataset.

```
model = models.Sequential([
    layers.Input((128, 128, 3)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(12, activation='softmax')
])
```

Com CNN do Zero

Vantagens observadas :

- Maior velocidade no processamento (treino e teste);
- Modelo bem mais leve;

Desvantagens :

- Modelo só trabalha bem com imagens similares aos exemplos do dataset

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 64)	36,928
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_3 (Conv2D)	(None, 12, 12, 64)	36,928
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 64)	589,888
dense_1 (Dense)	(None, 12)	780

Total params: 683,916 (2.61 MB)
Trainable params: 683,916 (2.61 MB)
Non-trainable params: 0 (0.00 B)

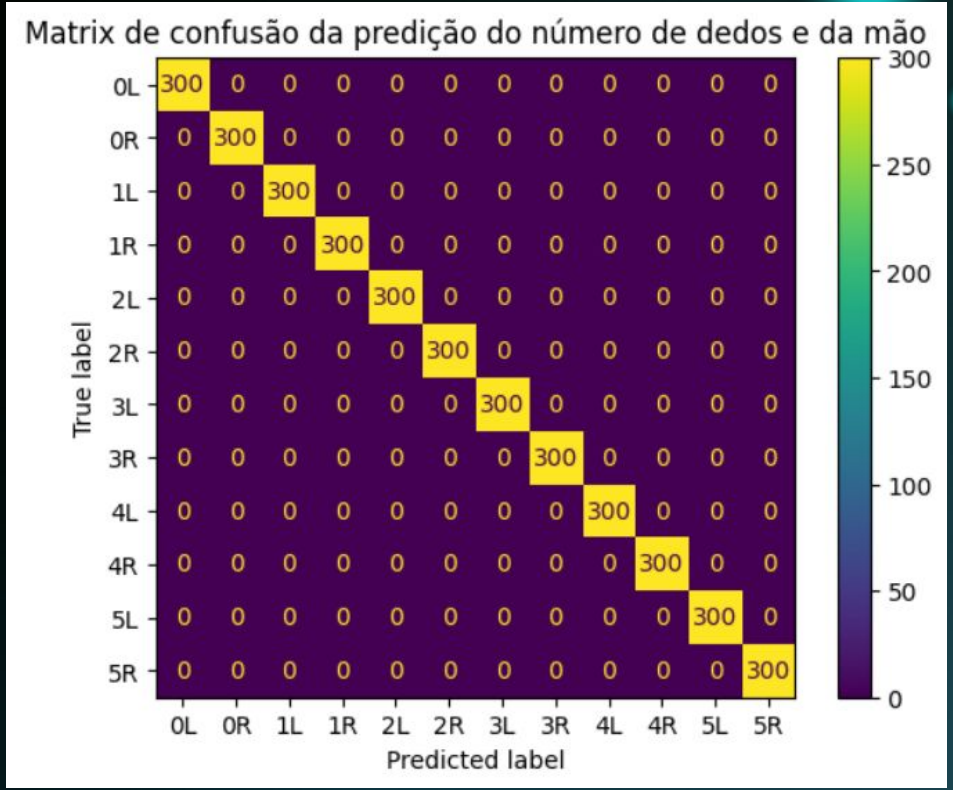
Com CNN do Zero - resultados de testes

Vantagens observadas :

- Maior velocidade no processamento (treino e teste);
- Modelo bem mais leve;

Desvantagens :

- Modelo só trabalha bem com imagens similares aos exemplos do dataset

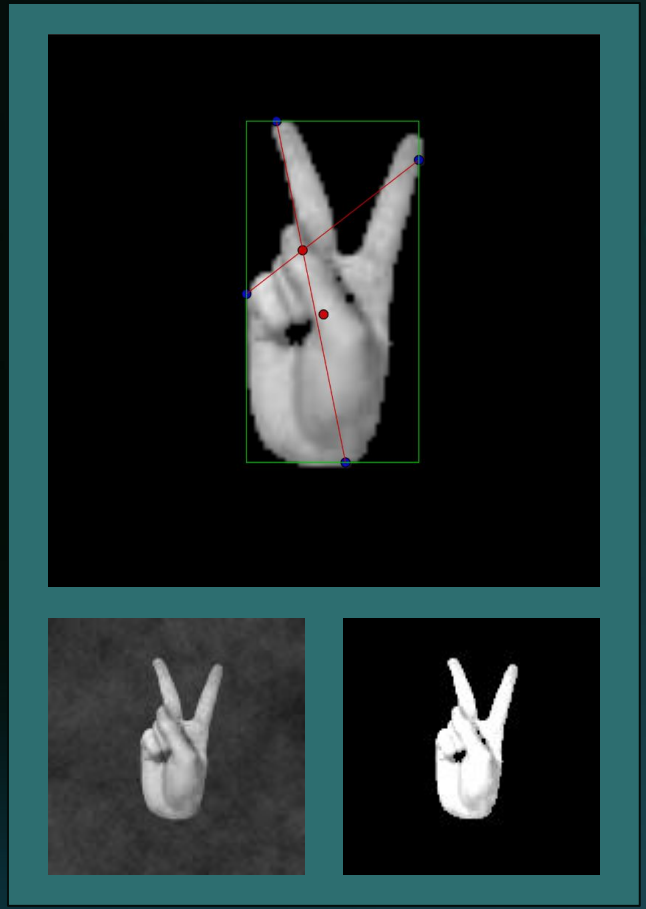


Com RandomForestClassifier do sklearn

Foi criada uma função que pega os parâmetros das imagens, calculando os pixels individualmente e pegando valores referentes a distribuição deles, densidade de pixels na área útil da imagem e ângulos entre pontos limites.

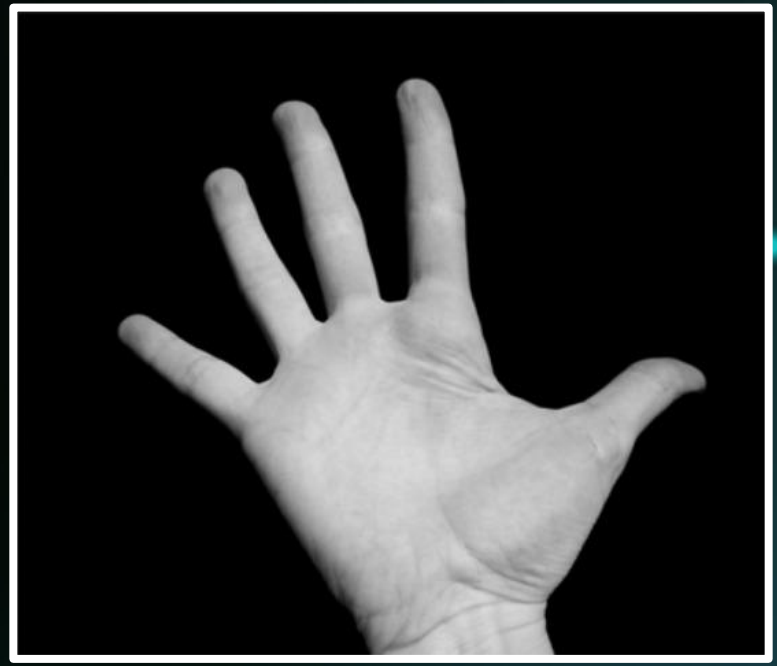
Após isso gerou-se um código para extrair esses dados de todas as imagens, salvando os valores em arquivos csv para o treinamento e para o teste.

Por fim, usou-se a biblioteca sklearn com o modelo Random Forest Classifier para ler os parâmetros extraídos das imagens.



Com RandomForestClassifier do sklearn

Passando a imagem ao lado que não está no dataset na função de processamento de imagem, obteve-se os parametros abaixo, que quando passados para a função de predição do modelo, gerou como resposta os valores verdadeiros (mão direita com 5 dedos).



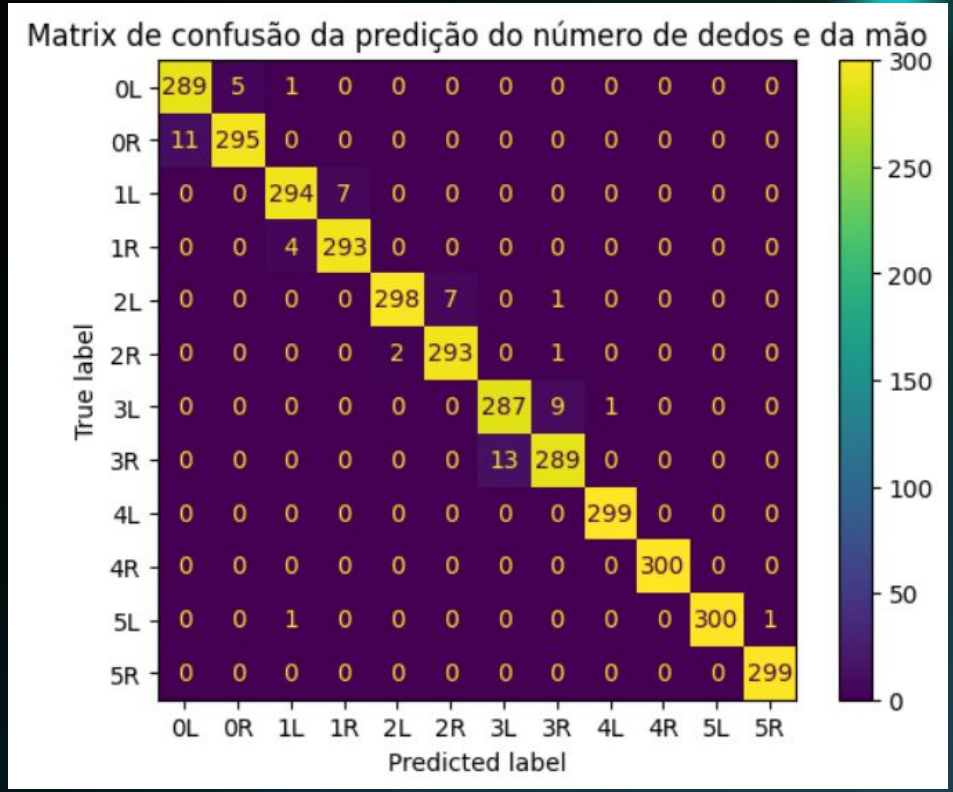
```
input = [0.3900642368384304, 0.0964737751825869, 1.3258176636680326, 0.49802775228195956, 0.5780498826891358, 49.7454
resp = model.predict([input])

print( HAND_MAP[ resp[0] ] )
```

```
mao: Direita, dedos: 5
```

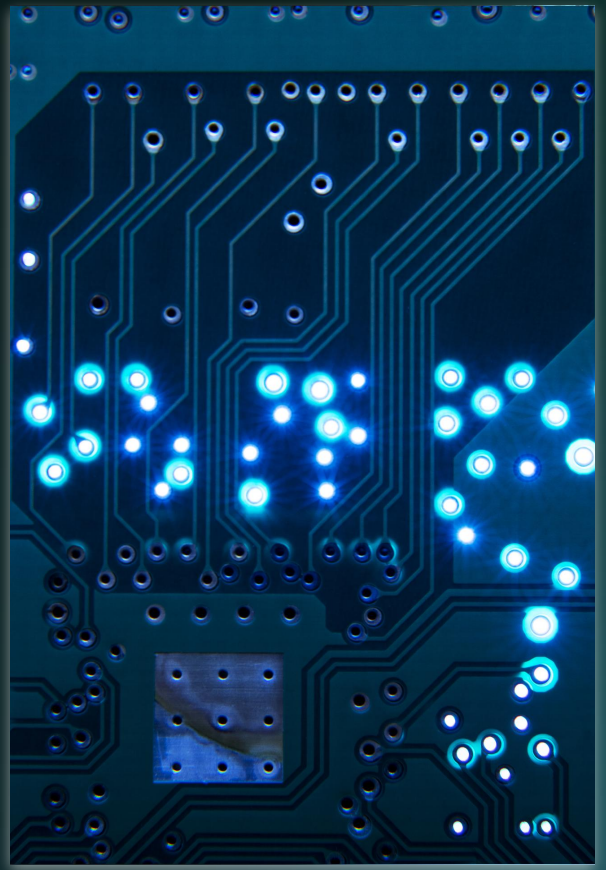
Com RandomForestClassifier do sklearn

Passando a imagem ao lado que não está no dataset na função de processamento de imagem, obteve-se os parametros abaixo, que quando passados para a fnção de predição do modelo, gerou como resposta os valores verdadeiros (mão direita com 5 dedos).

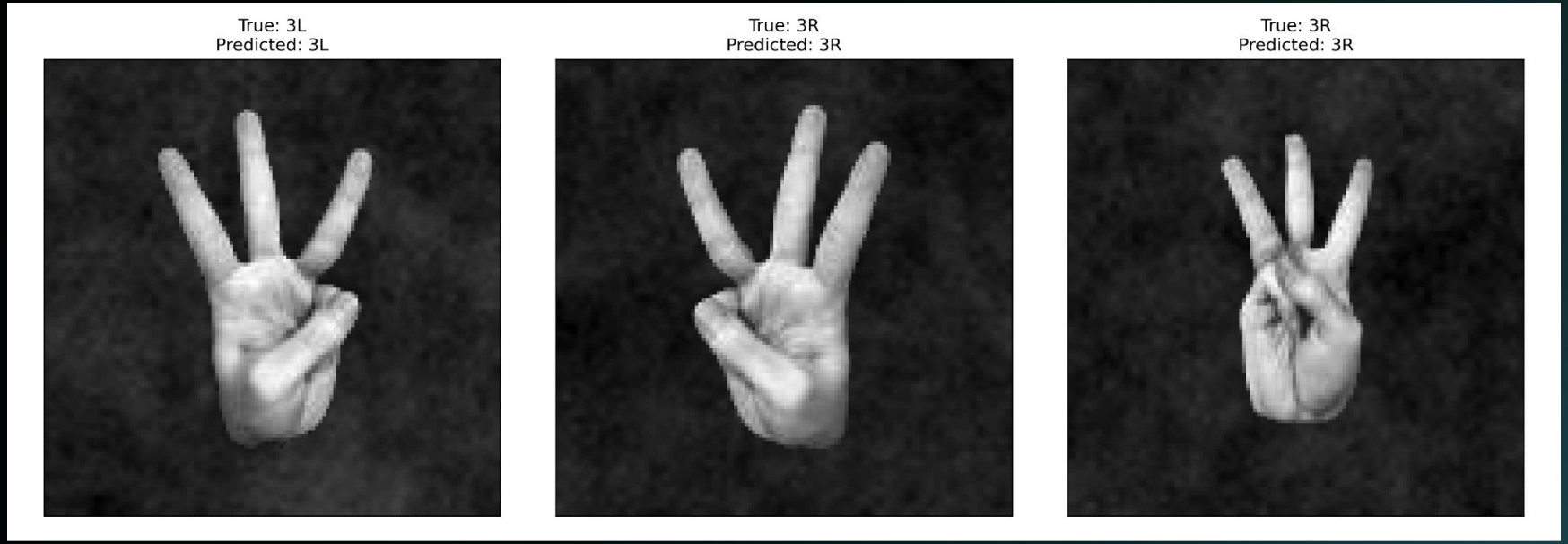


03

RESULTADOS _



Classificação das imagens



Classificação das imagens

Resultados

Versão do código	Accuracy	Loss	Tempo de execução
Modelo 99%	99%	2.9%	26 minutos
CNN do zero	99%	0.7%	389 segundos
RandomForest	98%	-	3.4 segundos

Modelos na prática

Salvando os modelos

Salvamento dos modelos após o treino e teste.

Os modelos foram salvos na pasta “Entrega 3”

```
model.save("../Entrega 3/cnn-zero.keras")
```

```
complete_model.save("../../Entrega 3/cnn-99.keras")
```

```
with open('../../Entrega 3/model_randomForest.pkl','wb') as f:
    pickle.dump(model,f)
```

0a523c62-73a2-48b8-a083-6aabb22cbb82_2R

0a564866-b4ba-44a5-a2fa-4b0797511e1d_3L

img-3R

img-hand-1L

img-hand-2R

img-hand-4L

img-teste-0R

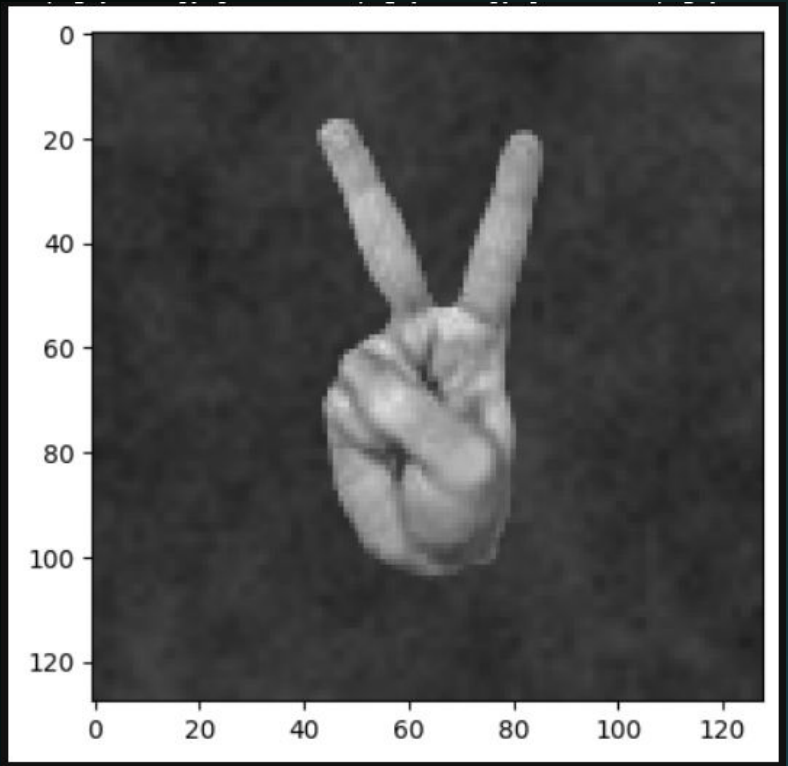
teste-3L

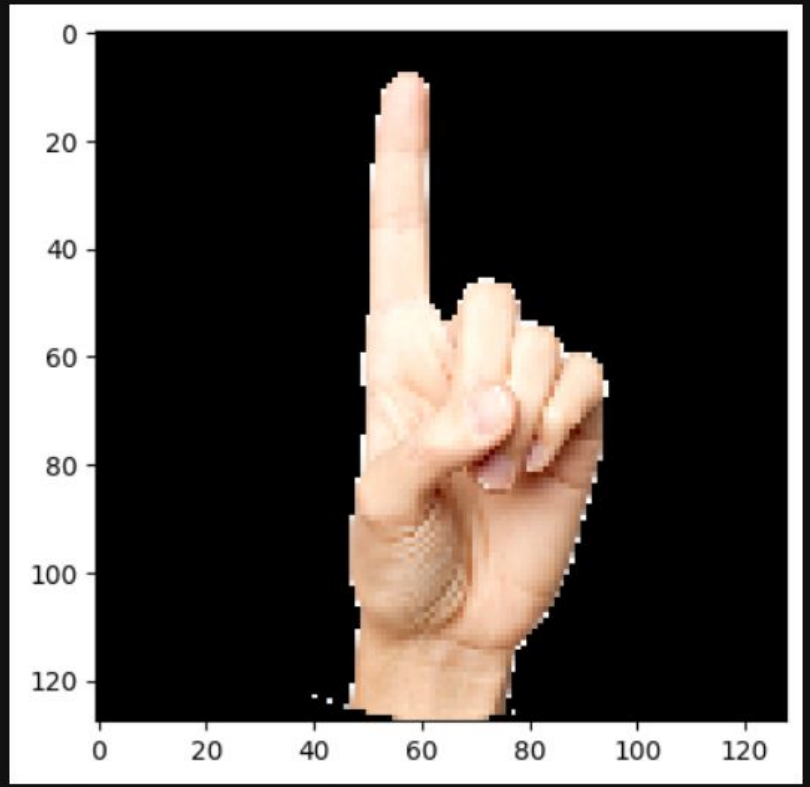
teste-3R

teste5R

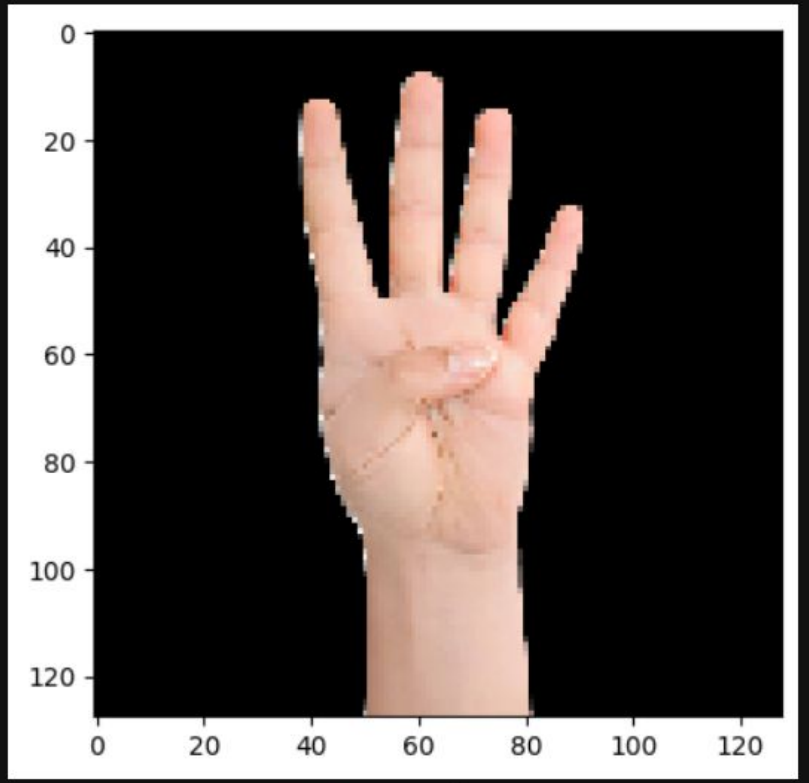
Carregando os modelos para serem facilmente executados

No arquivo Entrega 3/loadModelPipeline.ipynb os 3 modelos foram carregados e usados para verificar previsões com imagens diferentes do dataset

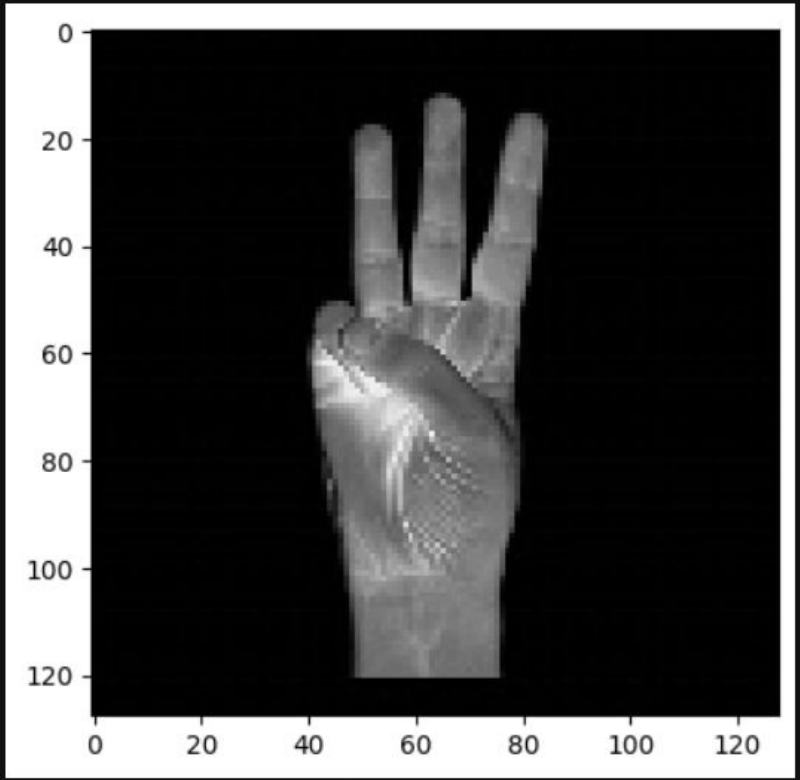




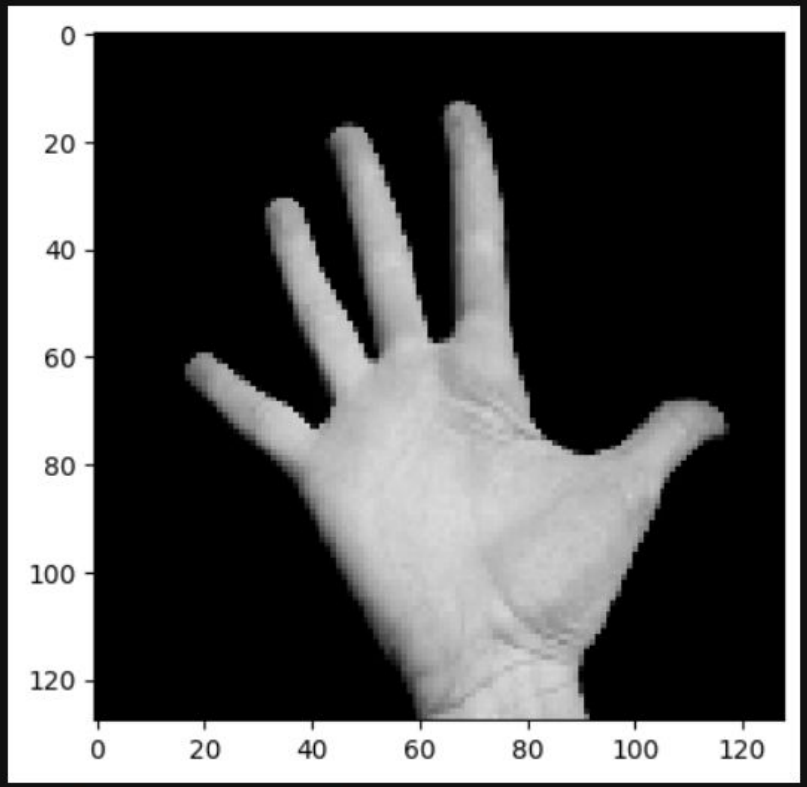
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Found 1 validated image filenames belonging to 1 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 146ms/step
Resposta -> 1L :
 * CNN___0 :  Previsto : 1R  -> Dedos : ✅ | Mão : ❌
 * CNN_99P :  Previsto : 1L  -> Dedos : ✅ | Mão : ✅
 * SEM_CNN :  Previsto : 2R  -> Dedos : ❌ | Mão : ❌
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
Found 1 validated image filenames belonging to 1 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step
Resposta -> 4L :
 * CNN___0 :  Previsto : 3L  -> Dedos : ❌ | Mão : ✅
 * CNN_99P :  Previsto : 3L  -> Dedos : ❌ | Mão : ✅
 * SEM_CNN :  Previsto : 4L  -> Dedos : ✅ | Mão : ✅
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
Found 1 validated image filenames belonging to 1 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
Resposta -> 3R :
 * CNN___0 :  Previsto : 3R  -> Dedos : ✅ | Mão : ✅
 * CNN_99P :  Previsto : 0L  -> Dedos : ❌ | Mão : ❌
 * SEM_CNN :  Previsto : 3R  -> Dedos : ✅ | Mão : ✅
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
Found 1 validated image filenames belonging to 1 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 178ms/step
Resposta -> 5R :
 * CNN___0 :  Previsto : 5R  -> Dedos : ✅ | Mão : ✅
 * CNN_99P :  Previsto : 5R  -> Dedos : ✅ | Mão : ✅
 * SEM_CNN :  Previsto : 5R  -> Dedos : ✅ | Mão : ✅
```

Estudantes

- Andreia Dourado
- Augusto Cesar
- Joao Victor
- Mario Umbelino
- Mauro Victor
- Rodrigo Chaveiro
- Victor Mendes

OBRIGADO!