Identificação e Classificação de Componentes de Interface

Visão Computacional Clássica
Deep Learning

https://github.com/magcid/visaoComputacional

Angélica Siqueira de Souza

Marcelo Fernando Rauber

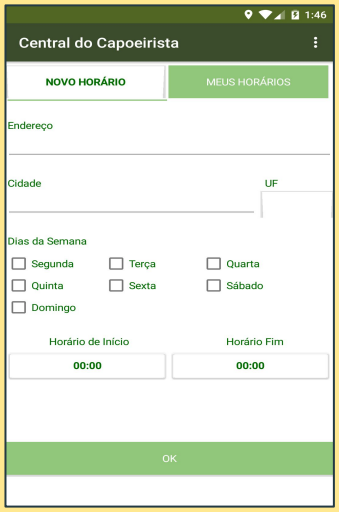
GQS
Grupo de Qualidade de Software

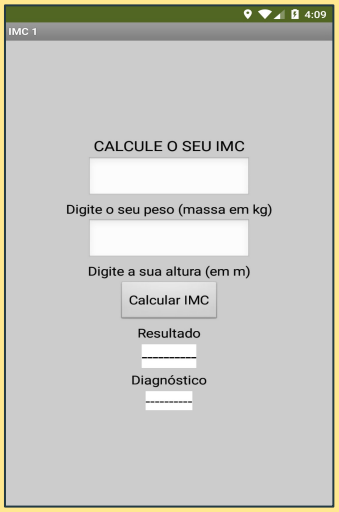
UNIVERSIDADE FEDERAL DE SANTA CATARINA

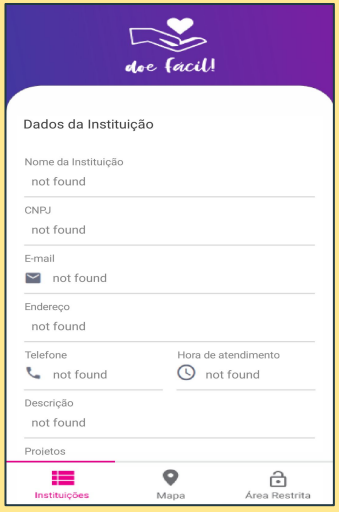
Objetivo

Detectar automaticamente os componentes de UI para identificar a posição, tamanho e o tipo do componente de UI nos screenshots de apps.

Exemplos de Screenshots de Apps da Galeria do App Inventor e do Contexto da Iniciativa Computação na Escola







Lista de Componentes de UI

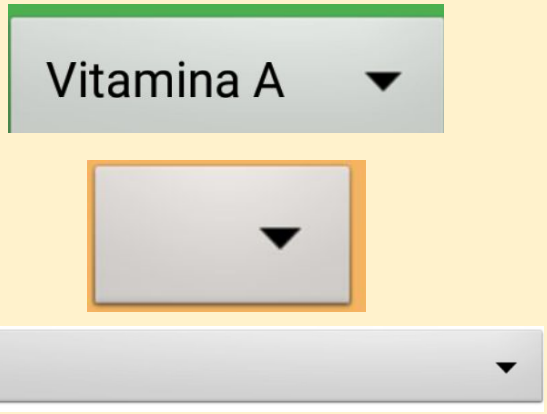
Button
CheckBox
DatePicker
Image
BackgroundImage
Label
ListPicker

ListView
PasswordTextBox
Slider
Spinner
Switch
TextBox

Notifier
TimePicker
WebViewer
Map
ImagePicker
VideoPlayer

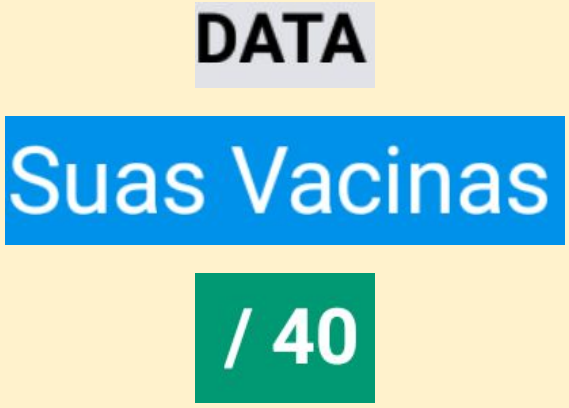
Exemplos dos Componentes de UI

Spinner

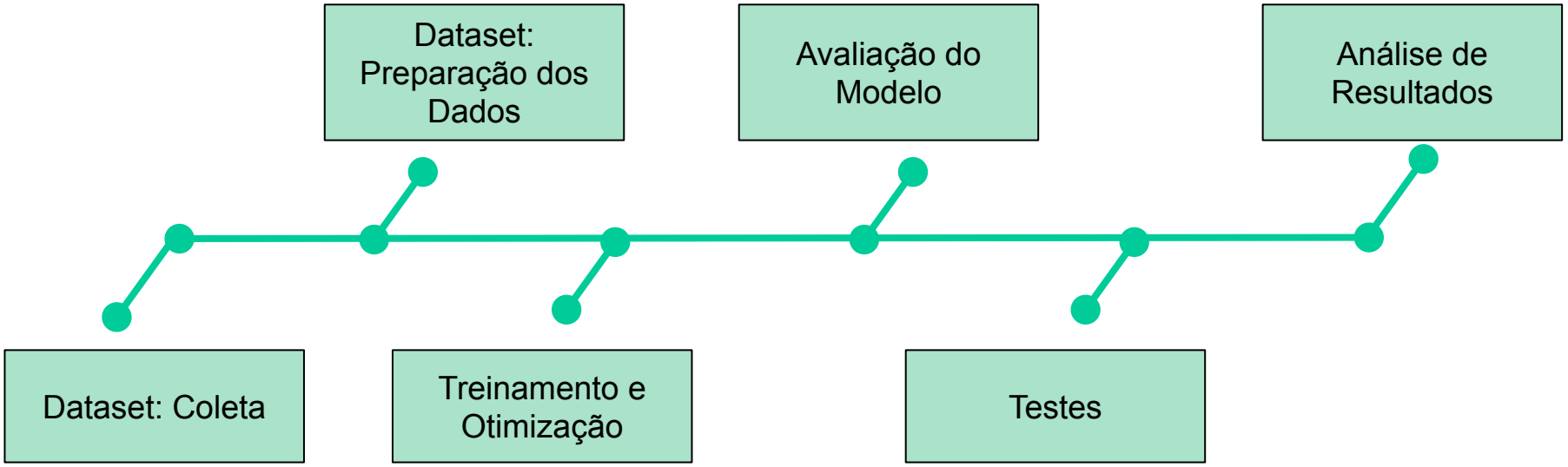


Button

Label



Etapas das Soluções



Dataset

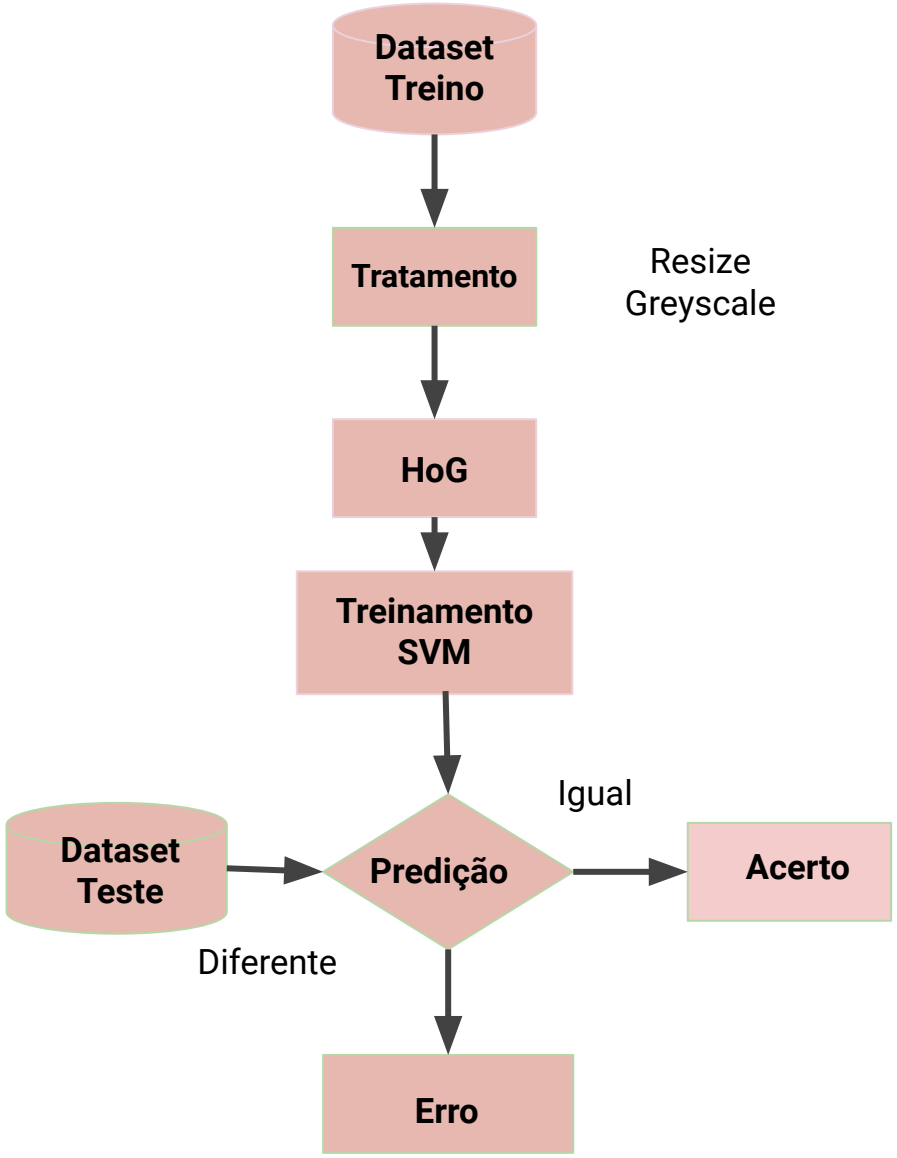
- Apps da iniciativa computação na escola.
- Apps da galeria App Inventor.

Total: 364 Screenshot (291 para treino e 73 para validação)

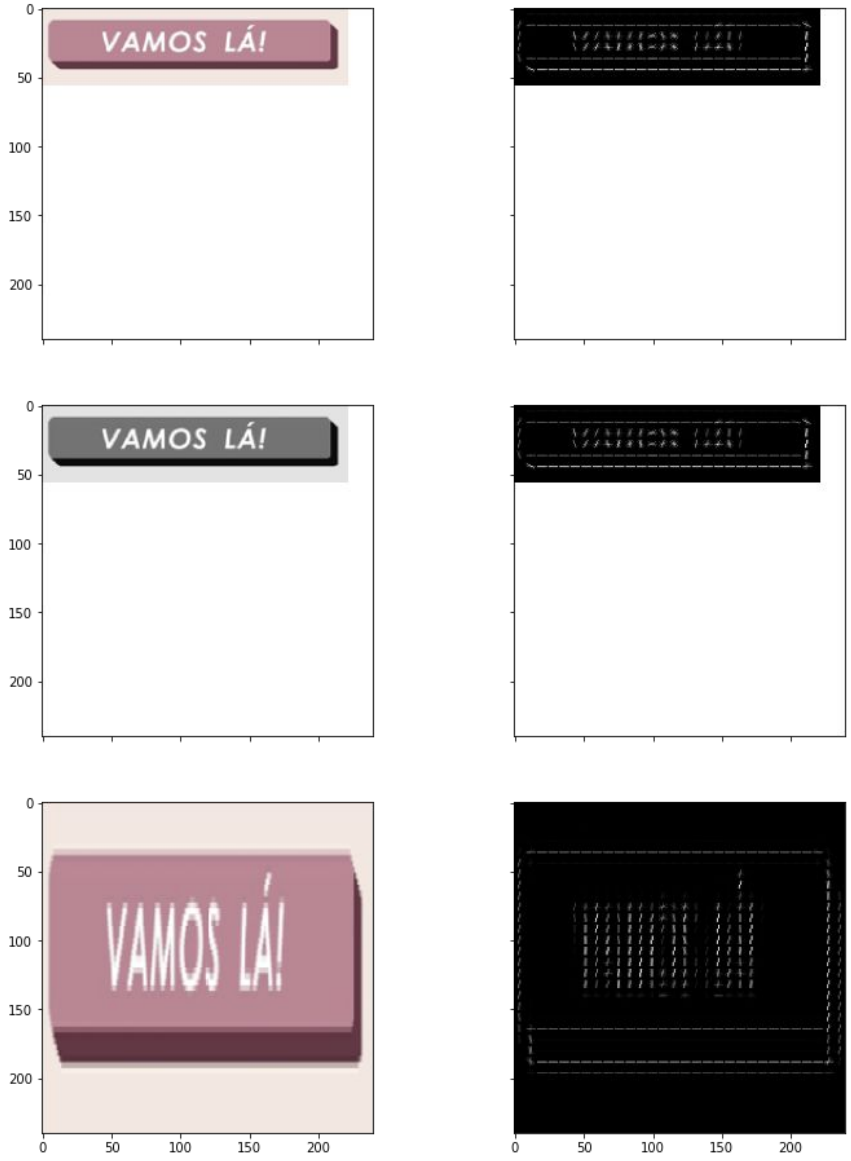
Visão Clássica - HOG

- Já prevendo uso de CNN, os rótulos marcados com LabelMe foram automaticamente recortados para novas imagens.
- As imagens foram redimensionadas, convertidas para escala de cinza
- Foi utilizado HOG (Histograma de Gradientes orientados) para as imagens.
- Foram criadas amostras para os componentes de UI de interesse
- Identificação foi feita com SVM - Suport Vector Machine (sklearn)

Visão Clássica - HOG

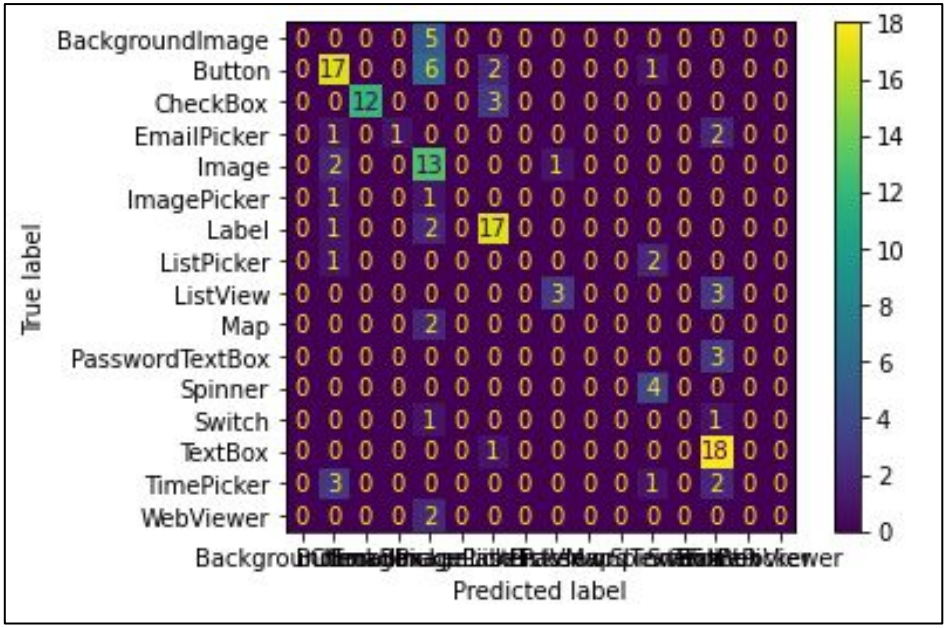


Visão Clássica - HOG



Visão Clássica - HOG - Resultado

	precision	recall	f1-score	support
BackgroundImage	0.00	0.00	0.00	5
Button	0.65	0.65	0.65	26
CheckBox	1.00	0.80	0.89	15
EmailPicker	1.00	0.25	0.40	4
Image	0.41	0.81	0.54	16
ImagePicker	0.00	0.00	0.00	2
Label	0.74	0.85	0.79	20
ListPicker	0.00	0.00	0.00	3
ListView	0.75	0.50	0.60	6
Map	0.00	0.00	0.00	2
PasswordTextBox	0.00	0.00	0.00	3
Spinner	0.50	1.00	0.67	4
Switch	0.00	0.00	0.00	2
TextBox	0.62	0.95	0.75	19
TimePicker	0.00	0.00	0.00	6
WebViewer	0.00	0.00	0.00	2
accuracy			0.63	135
macro avg	0.35	0.36	0.33	135
weighted avg	0.56	0.63	0.57	135

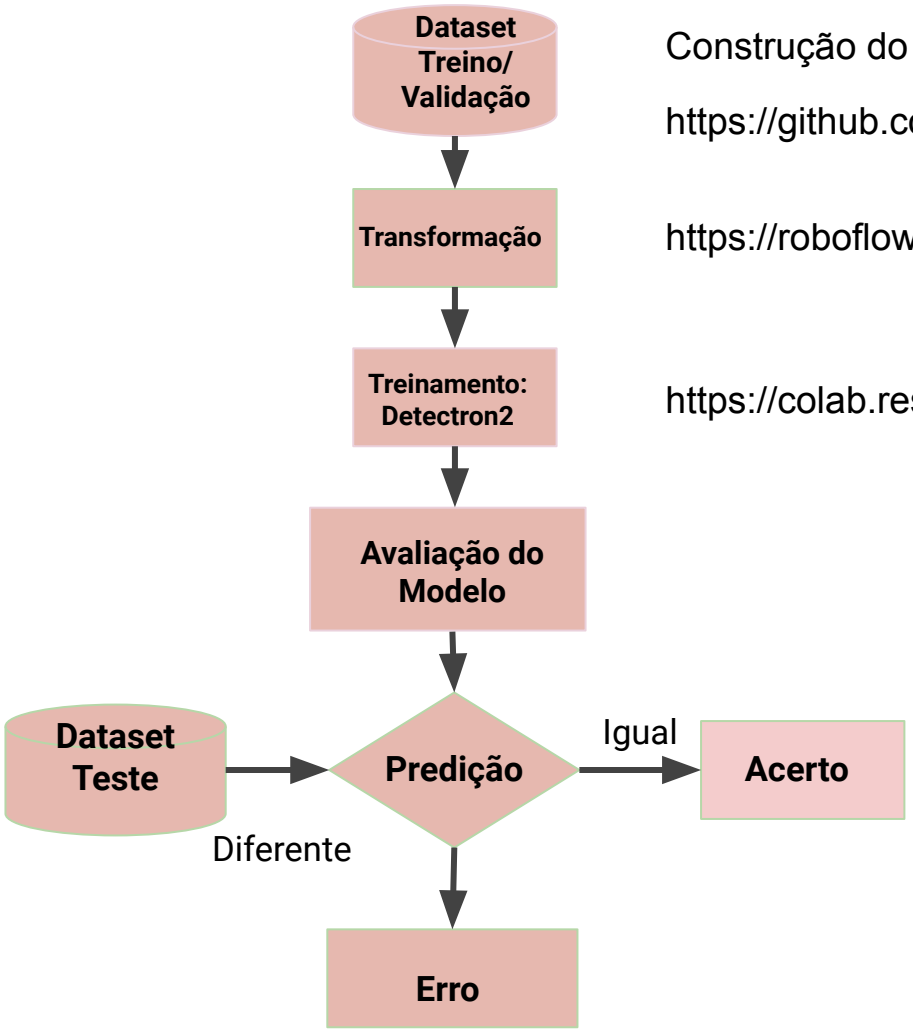


Deep Learning

https://github.com/facebookresearch/detectron2

- É uma biblioteca do Facebook AI Research's
- Se coloca como state-of-the-art para detecção e segmentação

Deep Learning - Detectron2



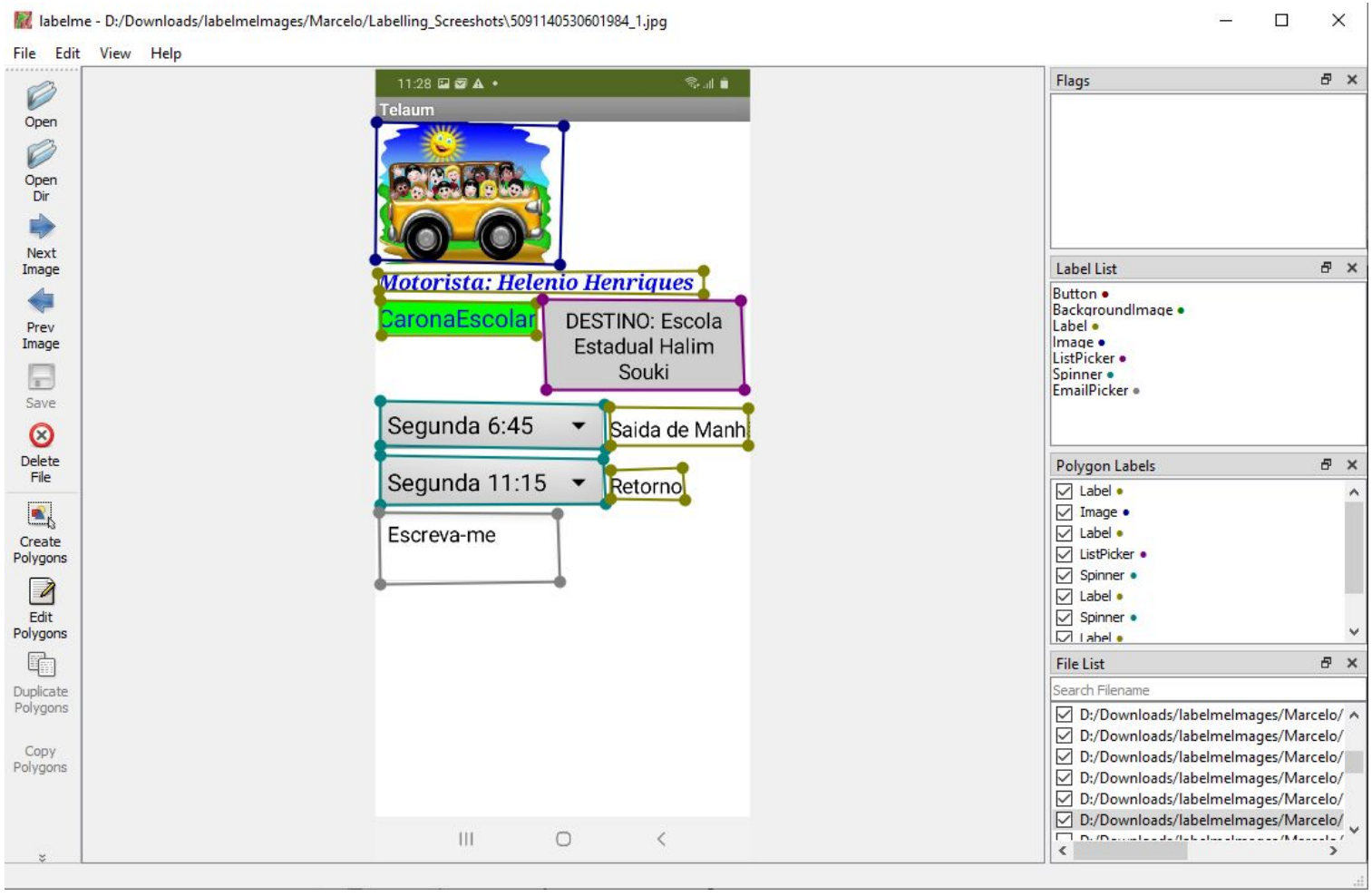
Construção do Dataset: Screenshot das Imagens

https://github.com/wkentaro/labelme

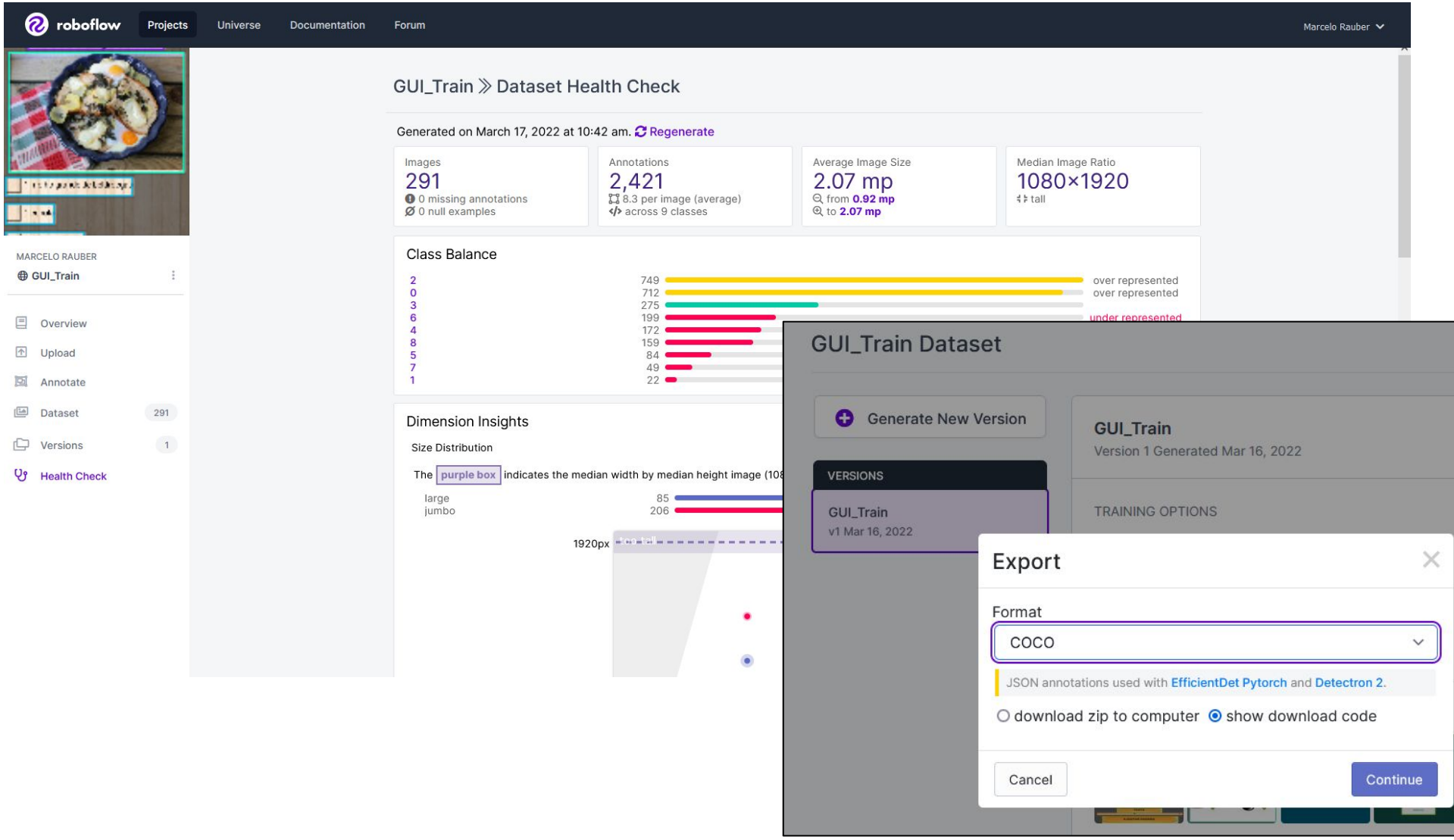
https://roboflow.com/

https://colab.research.google.com/

Deep Learning - Detectron2



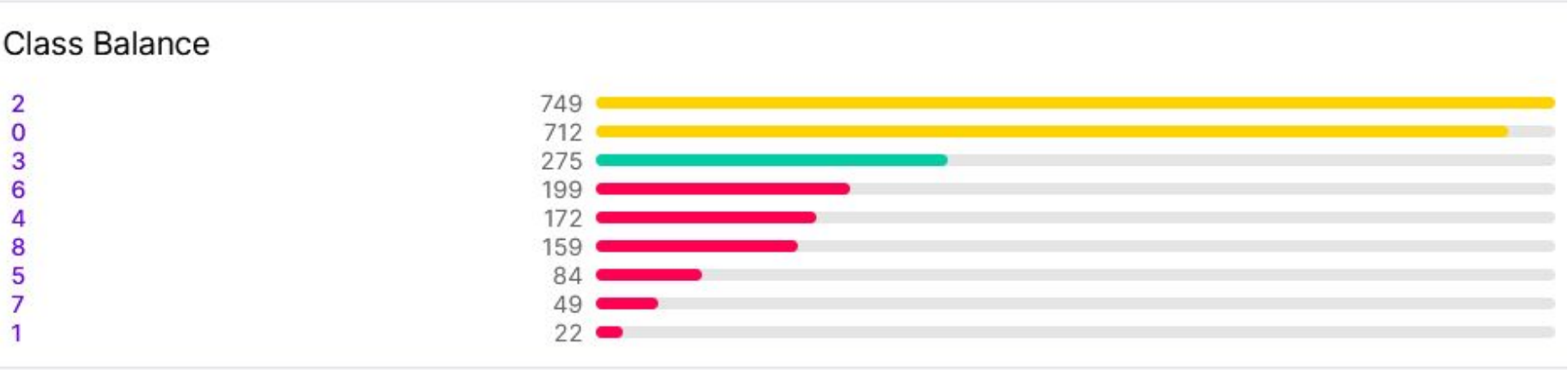
Deep Learning



Deep Learning

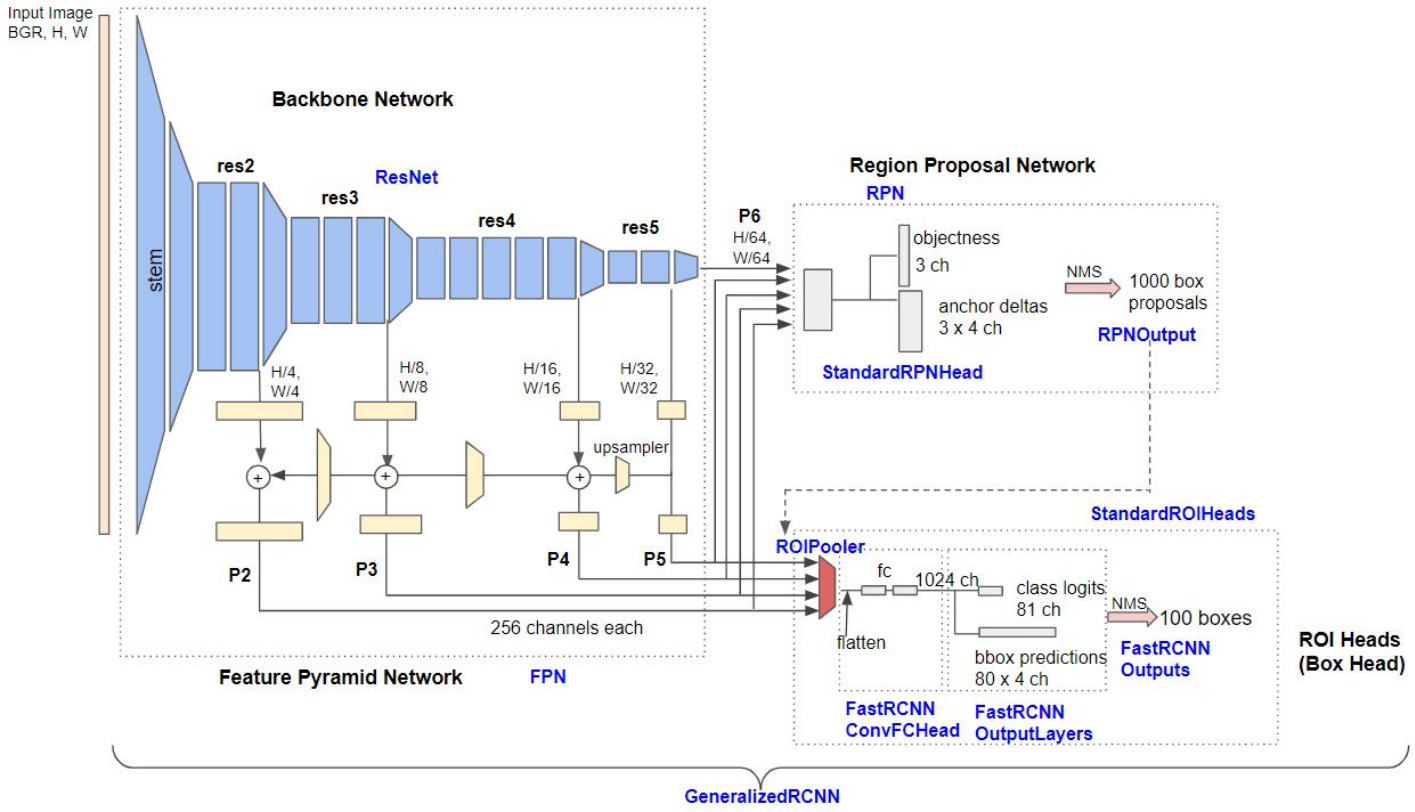
DataSet de Treinamento:

Images	Annotations	Average Image Size
291	2,421	2.07 mp
0 missing annotations	8.3 per image (average)	from **0.92 mp**
0 null examples	across 9 classes	to **2.07 mp**



Deep Learning - Detectron2

- CNN Redes Neurais Convolucionais
- Seleção do modelo: faster_rcnn_X_101_32x8d_FPN_3x



Detailed architecture of Base-RCNN-FPN. Source: https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd

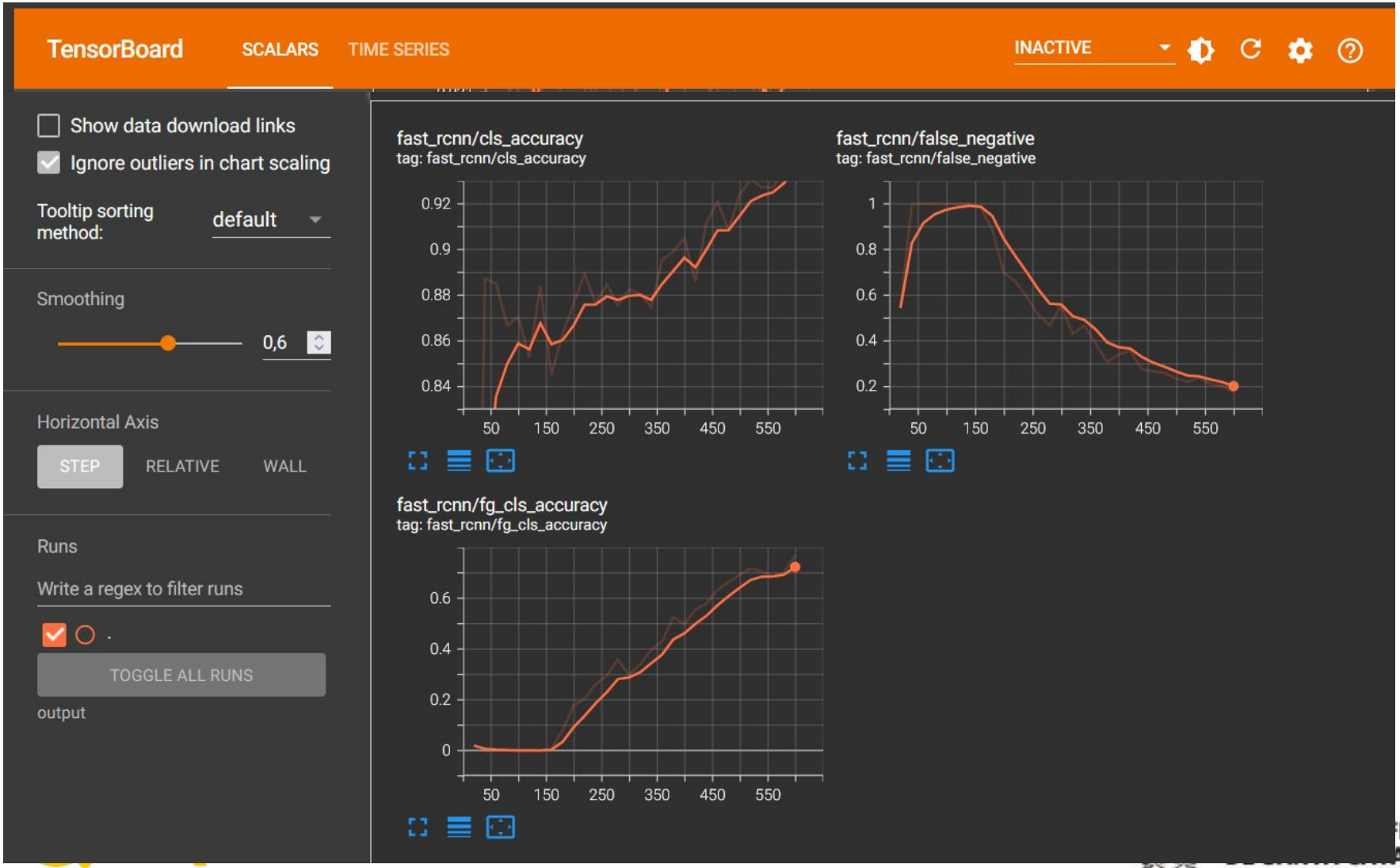
Deep Learning - Detectron2

- Aprendizado por transferência, 600 épocas.

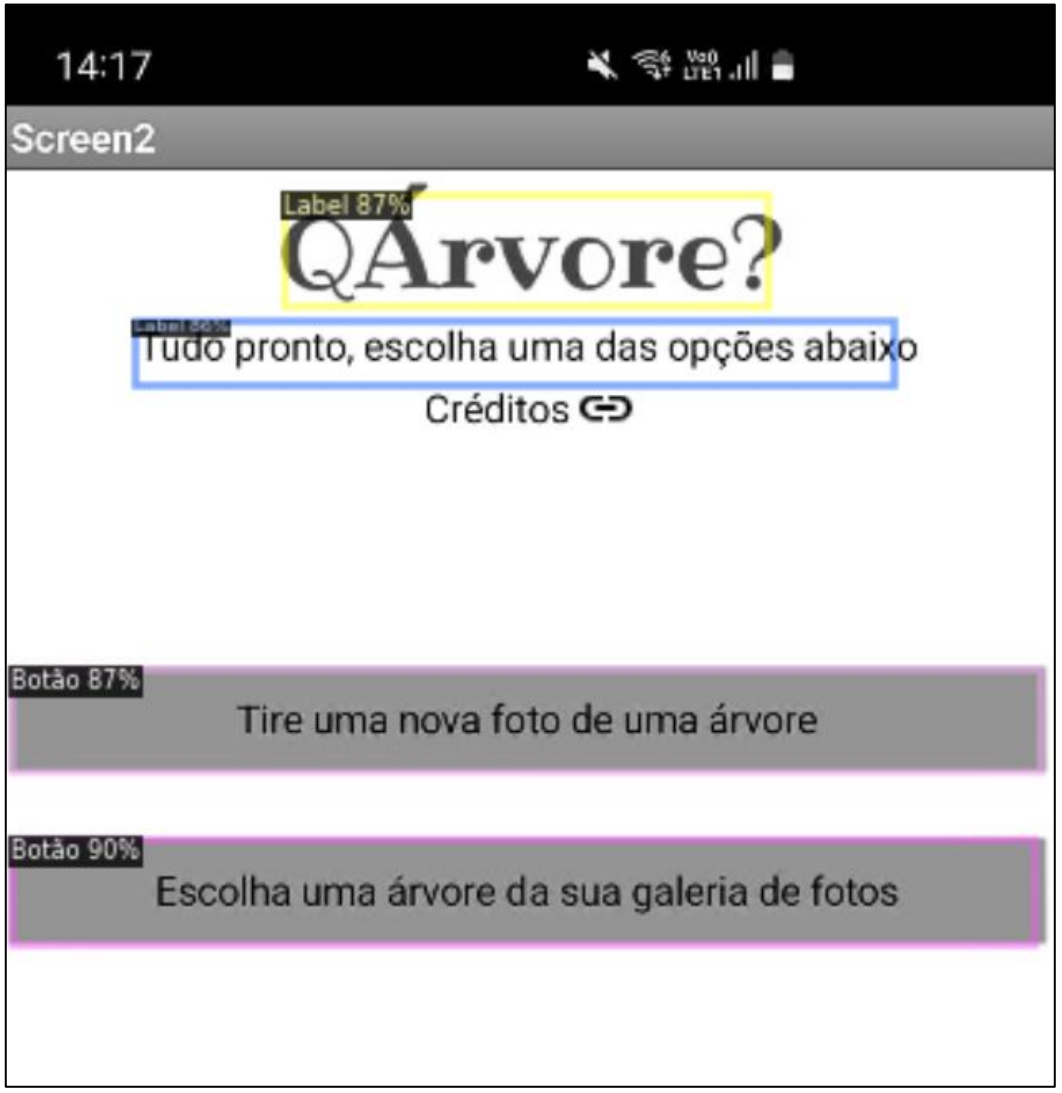
```
 eta: 0:24:46  iter: 179  total_loss: 1.211  loss_cls: 0.4711  loss_box_reg: 0.4964  loss_rpn_cls: 0.0372
 eta: 0:23:33  iter: 199  total_loss: 1.14  loss_cls: 0.4345  loss_box_reg: 0.4705  loss_rpn_cls: 0.02849  loss_rpn_loc: 0.1916
 eta: 0:22:21  iter: 219  total_loss: 1.008  loss_cls: 0.397  loss_box_reg: 0.4356  loss_rpn_cls: 0.02955  loss_rpn_loc: 0.1836
 eta: 0:21:09  iter: 239  total_loss: 1.103  loss_cls: 0.4057  loss_box_reg: 0.4427  loss_rpn_cls: 0.0303  loss_rpn_loc: 0.1994
 eta: 0:19:56  iter: 259  total_loss: 1.014  loss_cls: 0.3849  loss_box_reg: 0.4114  loss_rpn_cls: 0.0278  loss_rpn_loc: 0.1641
 eta: 0:18:46  iter: 279  total_loss: 1.024  loss_cls: 0.4048  loss_box_reg: 0.4186  loss_rpn_cls: 0.02784  loss_rpn_loc: 0.1694
 eta: 0:17:36  iter: 299  total_loss: 0.9285  loss_cls: 0.3662  loss_box_reg: 0.3484  loss_rpn_cls: 0.03013  loss_rpn_loc: 0.1917
 eta: 0:16:25  iter: 319  total_loss: 0.9163  loss_cls: 0.3677  loss_box_reg: 0.3497  loss_rpn_cls: 0.02462  loss_rpn_loc: 0.1726
 eta: 0:15:15  iter: 339  total_loss: 0.968  loss_cls: 0.3842  loss_box_reg: 0.363  loss_rpn_cls: 0.02773  loss_rpn_loc: 0.202  t
 eta: 0:14:04  iter: 359  total_loss: 0.8399  loss_cls: 0.3252  loss_box_reg: 0.3124  loss_rpn_cls: 0.01972  loss_rpn_loc: 0.1654
 eta: 0:12:54  iter: 379  total_loss: 0.827  loss_cls: 0.3052  loss_box_reg: 0.3234  loss_rpn_cls: 0.0168  loss_rpn_loc: 0.175  t
 eta: 0:11:44  iter: 399  total_loss: 0.7641  loss_cls: 0.2853  loss_box_reg: 0.2863  loss_rpn_cls: 0.02341  loss_rpn_loc: 0.1641
 eta: 0:10:33  iter: 419  total_loss: 0.8657  loss_cls: 0.3672  loss_box_reg: 0.3191  loss_rpn_cls: 0.02774  loss_rpn_loc: 0.1881
 eta: 0:09:23  iter: 439  total_loss: 0.6819  loss_cls: 0.2576  loss_box_reg: 0.265  loss_rpn_cls: 0.01737  loss_rpn_loc: 0.1504
 eta: 0:08:13  iter: 459  total_loss: 0.642  loss_cls: 0.2378  loss_box_reg: 0.2442  loss_rpn_cls: 0.01447  loss_rpn_loc: 0.1335
 eta: 0:07:02  iter: 479  total_loss: 0.7116  loss_cls: 0.2537  loss_box_reg: 0.2822  loss_rpn_cls: 0.01981  loss_rpn_loc: 0.163
 eta: 0:05:52  iter: 499  total_loss: 0.6882  loss_cls: 0.2287  loss_box_reg: 0.2733  loss_rpn_cls: 0.01539  loss_rpn_loc: 0.1728
 eta: 0:04:41  iter: 519  total_loss: 0.5978  loss_cls: 0.1878  loss_box_reg: 0.2278  loss_rpn_cls: 0.0123  loss_rpn_loc: 0.1647
 eta: 0:03:31  iter: 539  total_loss: 0.6473  loss_cls: 0.2073  loss_box_reg: 0.2493  loss_rpn_cls: 0.01473  loss_rpn_loc: 0.1619
 eta: 0:02:20  iter: 559  total_loss: 0.6609  loss_cls: 0.218  loss_box_reg: 0.2668  loss_rpn_cls: 0.01428  loss_rpn_loc: 0.131
 eta: 0:01:10  iter: 579  total_loss: 0.5194  loss_cls: 0.1848  loss_box_reg: 0.206  loss_rpn_cls: 0.01794  loss_rpn_loc: 0.1401
 eta: 0:00:00  iter: 599  total_loss: 0.5108  loss_cls: 0.1505  loss_box_reg: 0.2085  loss_rpn_cls: 0.01573  loss_rpn_loc: 0.1528
Overall training speed: 598 iterations in 0:36:59 (3.7108 s / it)
Total training time: 0:37:01 (0:00:02 on hooks)
```

Deep Learning - Detectron2

- Aprendizado por transferência, 600 épocas.



Deep Learning - Detectron2 - Resultado



Resultado - Validação

Para análise e validação dos resultados: CocoEvaluator
Daset distinto do treinamento.

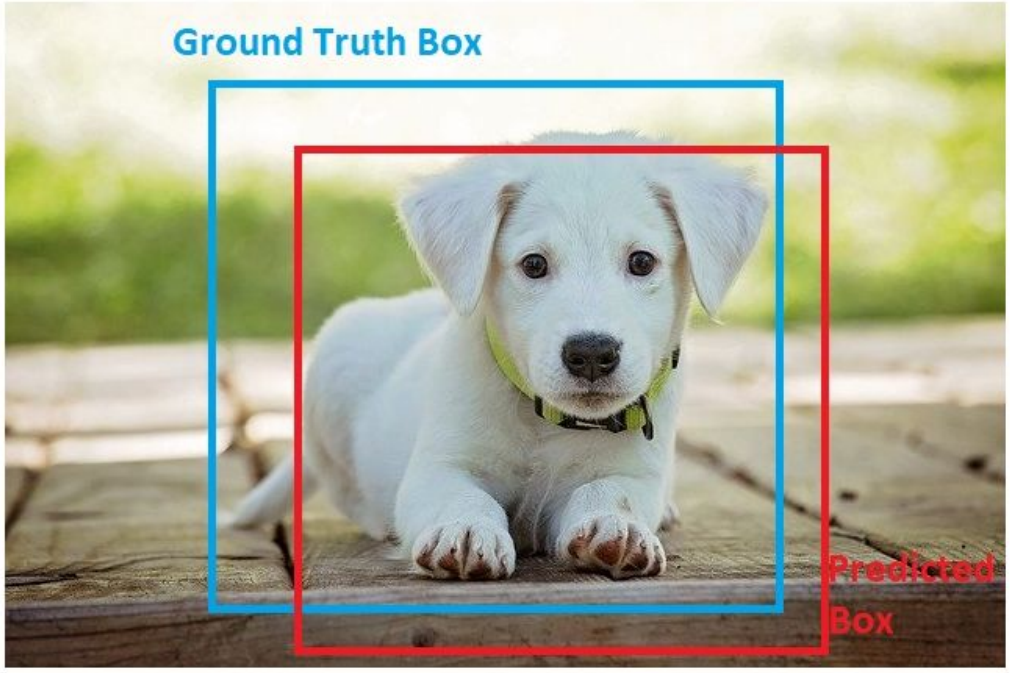


Figure 1. Ground truth bounding box and predicted bounding box in object detection.

2. Metrics

The following 12 metrics are used for characterizing the performance of an object detector on COCO:

```
Average Precision (AP):
  AP                    % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50            % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75            % AP at IoU=.75 (strict metric)
AP Across Scales:
  AP^small              % AP for small objects: area < 32^2
  AP^medium             % AP for medium objects: 32^2 < area < 96^2
  AP^large              % AP for large objects: area > 96^2
Average Recall (AR):
  AR^max=1              % AR given 1 detection per image
  AR^max=10             % AR given 10 detections per image
  AR^max=100            % AR given 100 detections per image
AR Across Scales:
  AR^small              % AR for small objects: area < 32^2
  AR^medium             % AR for medium objects: 32^2 < area < 96^2
  AR^large              % AR for large objects: area > 96^2
```

https://debuggercafe.com/evaluation-metrics-for-object-detection/
https://cocodataset.org/#detection-eval

Resultado - Validação

```
[03/17 18:05:07 d2.data.datasets.coco]: Loaded 73 images in COCO format from ./valid/_annotations.coco.json
[03/17 18:05:07 d2.data.build]: Distribution of instances among all 10 categories:
|   category    | #instances   |  category  | #instances   |  category  | #instances   |
|:-------------:|:-------------|:----------:|:-------------|:----------:|:-------------|
| GUI-compone.. | 0            |   Botão    | 168          |    Mapa    | 5            |
|     Label     | 224          |  TextBox   | 99           |   Imagem   | 41           |
|    Slider     | 21           |  CheckBox  | 32           |   Switch   | 15           |
|  ListPicker   | 35           |            |              |            |              |
|     total     | 640          |            |              |            |              |
```

```
[03/17 18:05:22 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 30.374 | 49.073 | 32.635 | 18.058 | 33.733 | 24.725 |
[03/17 18:05:22 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category       | AP     | category   | AP     | category   | AP     |
|:---------------|:-------|:-----------|:-------|:-----------|:-------|
| GUI-components | nan    | Botão      | 61.935 | Mapa       | 0.000  |
| Label          | 36.480 | TextBox    | 39.052 | Imagem     | 30.200 |
| Slider         | 30.309 | CheckBox   | 59.402 | Switch     | 15.986 |
| ListPicker     | 0.000  |            |        |            |        |
OrderedDict([('bbox', {'AP': 30.373853611965725, 'AP50': 49.07306140599386, 'A
```

```
[03/17 18:05:22 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.05 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.304
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.491
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.326
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.181
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.337
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.247
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.134
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.344
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.353
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.213
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.377
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.305
```

Average Precision (AP)

Conclusão

- A utilização de Faster r-CNN levou a resultados satisfatórios com Deep Learning, sendo que o conjunto de treinamento atingiu 93% de acurácia.
- O conjunto de validação de dados mostrou uma precisão média de 30,37% seguindo o padrão CocoEvaluator;
- Nos experimentos a utilização da abordagem clássica mostrou-se viável para classificar os componentes individuais, mas é difícil identificação dos componentes (potencial trabalho futuro);
- Trabalho futuros e melhorias:
  - Testar com outros modelos CNN, visando comparar e melhorar a precisão
  - Ampliação e melhoria do dataset e da sua rotulação pode levar a melhores resultados.

Obrigado!

Identificação e Classificação de Componentes de Interface

Visão Computacional Clássica
Deep Learning

https://github.com/magcid/visaoComputacional

Angélica Siqueira de Souza

Marcelo Fernando Rauber