

Departamento de electrónica, telecomunicações e informática

Curso 8204 - Mestrado Integrado em Engenharia Electrónica e Telecomunicações
Ano Lectivo 2019/2020

Recuperação e identificação de momentos em imagens/vídeos

Ponto de situação

Autor:

76587 Júlio Miguel Braz da Costa Silva

Data 20 de Dezembro de 2019

Orientador António Neves

Co-Orientador Ricardo Ribeiro

Conteúdo

1	Objetivos propostos na reunião 12/11/2019	2
2	Programa criado para deteção baseado em inputs do utilizador	2
2.1	Descrição do programa	2
2.2	Resultados obtidos	2
3	Wordclouds	4
3.1	Descrição do programa criado para gerar as wordclouds.	4
3.2	Resultados obtidos	5
3.3	Análise das wordclouds obtidas	7
4	State-of-the-art	7
4.1	Single Shot Detection	7
4.1.1	Resultados obtidos	7
4.2	CornerNet Lite	8
4.2.1	Resultados obtidos	9
4.3	Cascade Mask R-CNN	9
4.4	Google Cloud Vision API	10
5	Escrita do capítulo	11
6	Bibliografia	12

1 Objetivos propostos na reunião 12/11/2019

- Geração de anotação de todas as imagens. - **Concluído.**
- Guardar a anotação num ficheiro json/txt. - **Concluído**
- Criação de uma programa que detete uma determinada keyword (com uma percentagem mínima de erro associada) e que devolva todas as imagens, num visualizador, em que essa mesma keyword foi identificada. - **Concluído.**
- Criação de wordclouds, usando imagens idênticas, para redes diferentes. - **Concluído**
- Pesquisa de state-of-the-art mais recente - **Concluído**
- Testes com novas redes/APIs - **Não Concluído** - Apenas foi possível testar 1 rede nova. Mais detalhes na secção de state-of-the-art.
- Escrita de capítulo machine learning/state-of-the-art/algoritmos disponíveis/bibliotecas disponíveis/ etc - **Em desenvolvimento**

2 Programa criado para deteção baseado em inputs do utilizador

2.1 Descrição do programa

O algoritmo criado utiliza a rede neural YoloV3 e permite identificar até 80 keywords diferentes. Recebe dois inputs sendo eles : keyword a identificar e a percentagem minima de certeza. Após isto, o algoritmo corre a deteção numa pasta com várias imagens e para todas as imagens que cumpram os critérios anteriormente definidos, é gerado um ficheiro json que guarda as bounding boxes, a keyword e a percentagem. Com a conclusão da deteção, o algoritmo guarda todas as imagens identificadas numa pasta á parte e abre todas as imagens dessa mesma pasta de forma a visualizar as deteções realizadas.

2.2 Resultados obtidos

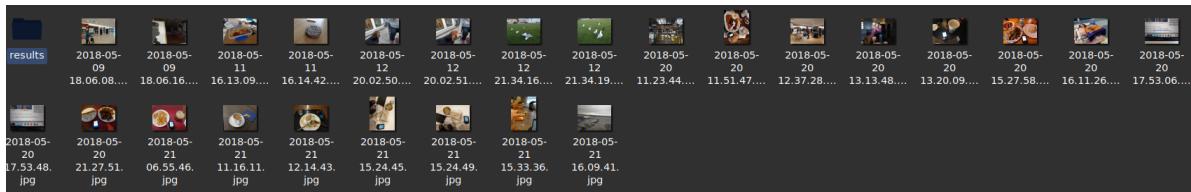


Figura 1: Pasta que o programa irá analisar.

Foram escolhidas apenas 25 imagens para fazer o teste de forma a diminuir o tempo que o programa demora a correr, no entanto este programa foi testado para pastas maiores e funcionou da mesma maneira mas mais lento.

```

-----MENU-----
All images in the directory :
['2018-05-09 18.06.08.jpg', '2018-05-11 16.14.42.jpg', '2018-05-20 11.23.44.jpg', '2018-05-20 21.27.51.jpg', '2018-05-20 17.53.48.jpg', '2018-05-11 16.13.09.jpg', 'results', '2018-05-12 20.02.51.jpg', '2018-05-20 16.11.26.jpg',
'2018-05-21 15.24.45.jpg', '2018-05-20 11.51.47.jpg', '2018-05-21 12.14.43.jpg', '2018-05-20 13.20.09.jpg', '2018-05-12 21.34.16.jpg', '2018-05-21 15.24.49.jpg', '2018-05-12 21.34.19.jpg', '2018-05-20 15.27.58.jpg', '2018-05-09 16.16.jpg',
'2018-05-20 13.13.48.jpg', '2018-05-21 06.55.46.jpg', '2018-05-21 15.33.36.jpg', '2018-05-21 16.09.41.jpg', '2018-05-12 21.11.16.jpg', '2018-05-20 20.02.50.jpg', '2018-05-20 12.37.28.jpg']

Words available for detection:
dict.keys(['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant',
'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair dryer', 'toothbrush'])

Choose a word for detection : notword
Please add a word that is available for detection:cup
You have chosen : cup
Choose a minimum percentage for the detection :1000000000
Please add a percentage between 0 and 100:50

```

Figura 2: Menu do algoritmo

Nesta imagem é possível observar o menu criado. Aqui o utilizador consegue ver todas as imagens que estão dentro da pasta e todas as palavras-chave que o algoritmo consegue identificar. O utilizador pode escolher a palavra-chave que quer identificar e qual a percentagem mínima de erro. No caso observado a palavra-chave escolhida foi "cup" com uma percentagem de erro de 50%.

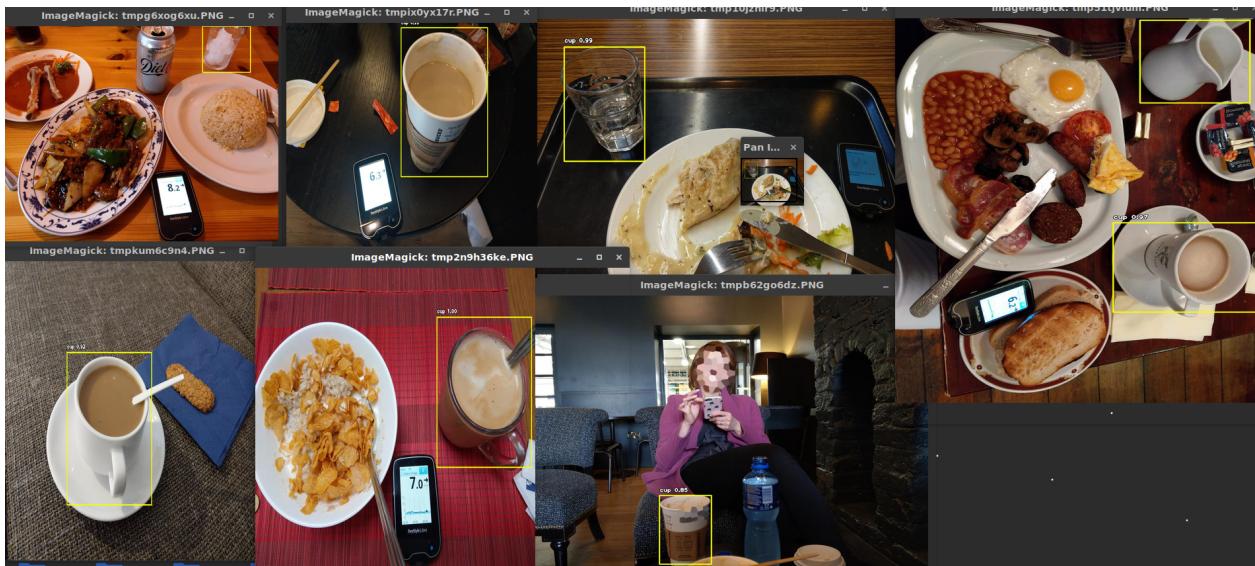


Figura 3: Deteções realizadas pelo algoritmo

Após correr o algoritmo, as imagens detetadas são automaticamente abertas. Observando a figura 3, é possível verificar que o algoritmo detetou 7 imagens que continham a keyword "cup" com uma percentagem de erro mínima de 50%.

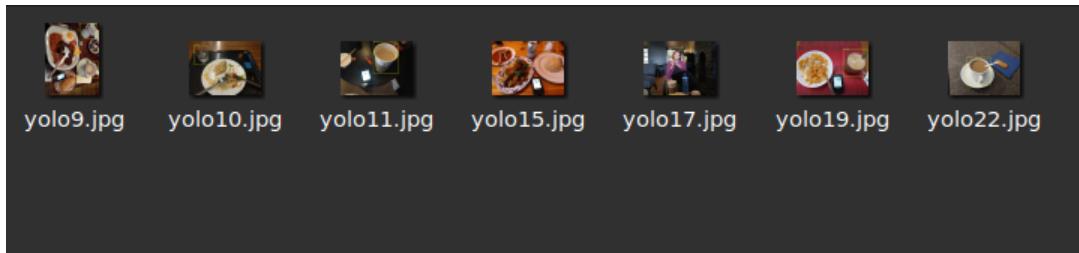


Figura 4: Pasta onde as imagens detetadas são guardadas

Na figura 4 é possível observar que as deteções foram guardadas corretamente.

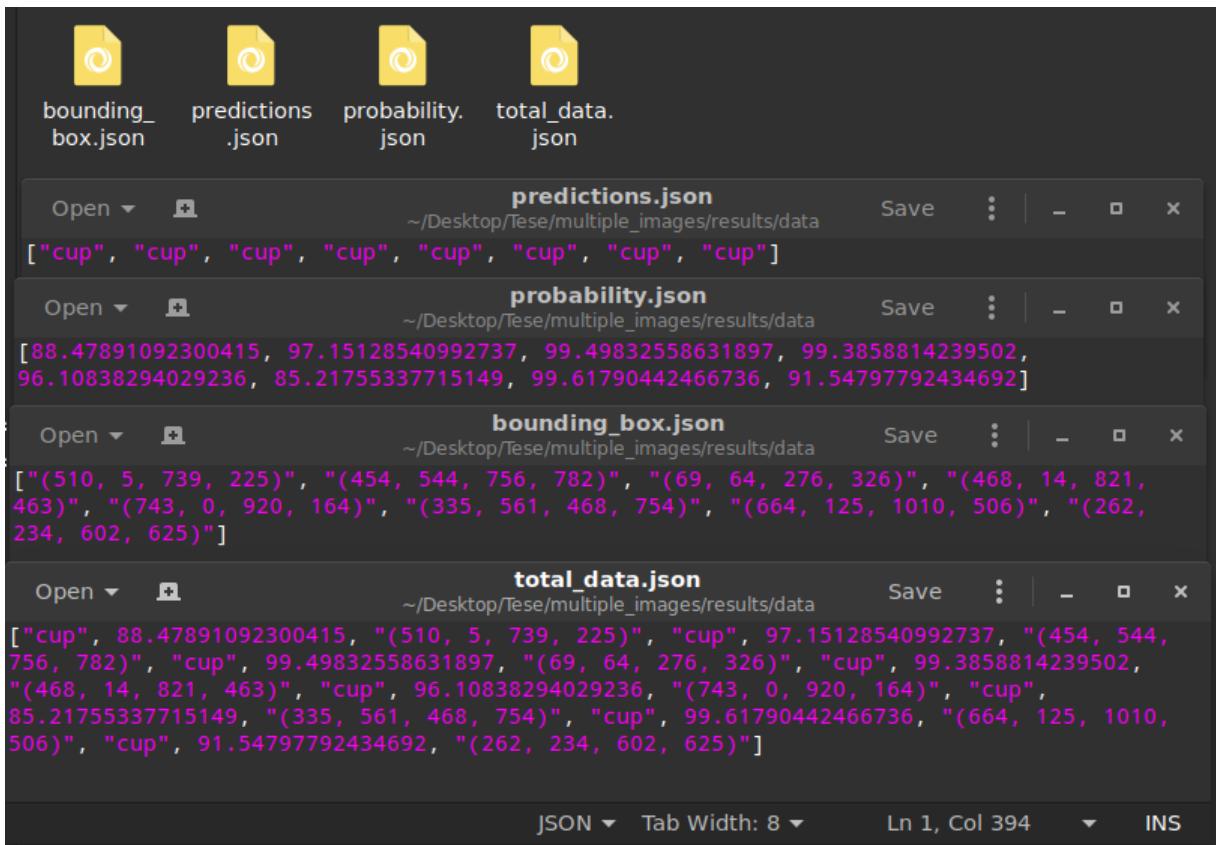


Figura 5: Json data

Assim que o programa conclui, 4 ficheiros em formato json são criados, tal como é possível verificar na imagem acima.

No ficheiro predictions.json são guardadas as deteções realizadas. Como apenas se fez uma procura por "cup" então somente "cup" irá aparecer nas keywords registadas no ficheiro json.

No ficheiro probability.json são guardadas as probabilidades de cada deteção de "cup", sendo que nenhuma destas será menor que 50, visto que foi o minímo admitido para este teste.

No ficheiro boundingbox.json são guardadas as localizações das bounding boxes para cada deteção realizada.

O ficheiro totaldata.json é simplesmente a junção dos últimos 3 ficheiros num só.

3 Wordclouds

3.1 Descrição do programa criado para gerar as wordclouds.

Para gerar as wordclouds foram utilizadas 3 redes neurais diferentes, sendo elas : RetinaNet, YoloV3, TinyYoloV3.

Foi desenvolvido um programa que corresse as 3 redes para o mesmo subset de imagens e guardasse todas as deteções para uma percentagem mínima de 30% num ficheiro json.

Após isto, foi utilizada uma ferramenta online para gerar as wordclouds com base nas deteções guardadas nos ficheiros json.



Figura 6: Imagens a serem analisadas pelas 3 redes

Na figura acima podemos ver o subset de imagens escolhidas para serem analisadas pelas 3 redes. As imagens foram todas escolhidas manualmente, de forma a serem idênticas no seu conteúdo.

3.2 Resultados obtidos

```

retina_data.json
tiny_yolo_data.json
yolo_data.json

```

```

retina_data.json
[{"bottle", "cup", "bowl", "dining table", "dining table", "cup", "sandwich", "dining table", "person", "bottle", "cup", "knife", "spoon", "fork", "hot dog", "sandwich", "knife", "knife", "fork", "fork", "bottle", "sandwich", "knife", "dining table", "cup", "sandwich", "knife", "cup", "cell phone", "dining table", "carrot", "person", "cup", "person", "dining table", "dining table", "bowl", "bowl", "cup", "bowl", "donut", "cell phone", "cup", "donut", "dining table", "spoon", "knife", "cup", "bowl", "cup", "spoon", "fork", "knife", "bottle", "cup", "fork", "knife", "pizza", "bowl", "cell phone", "sandwich", "dining table"}]

tiny_yolo_data.json
[{"bowl", "sandwich", "cup", "donut", "sandwich", "fork", "bottle", "bottle", "cell phone", "cake", "cup", "cell phone", "donut", "donut", "bowl", "cup", "cell phone", "fork", "cup", "bottle", "bottle"}]

yolo_data.json
[{"dining table", "bowl", "cup", "dining table", "sandwich", "cup", "dining table", "sandwich", "sandwich", "spoon", "knife", "knife", "fork", "bottle", "bottle", "cell phone", "dining table", "sandwich", "spoon", "cup", "cup", "dining table", "bowl", "cup", "cell phone", "dining table", "bowl", "bowl", "spoon", "cup", "cell phone", "dining table", "bowl", "bowl", "knife", "knife", "fork", "cup", "bottle", "bottle", "bottle"}]

```

Figura 7: Ficheiros json gerados

A Figura 7 mostra o conteúdo guardado nos ficheiros json, onde, para cada rede, é guardada cada anotação de cada deteção realizada.

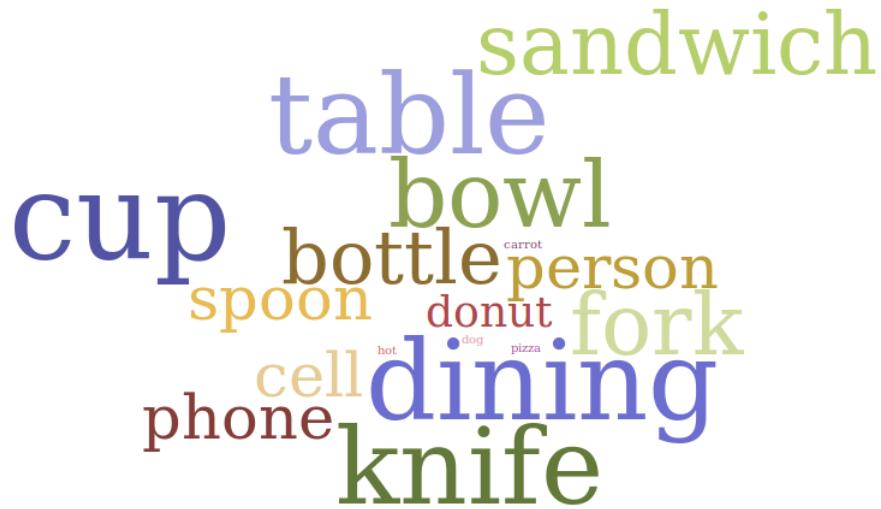


Figura 8: Wordcloud para as deteções da retinaNet

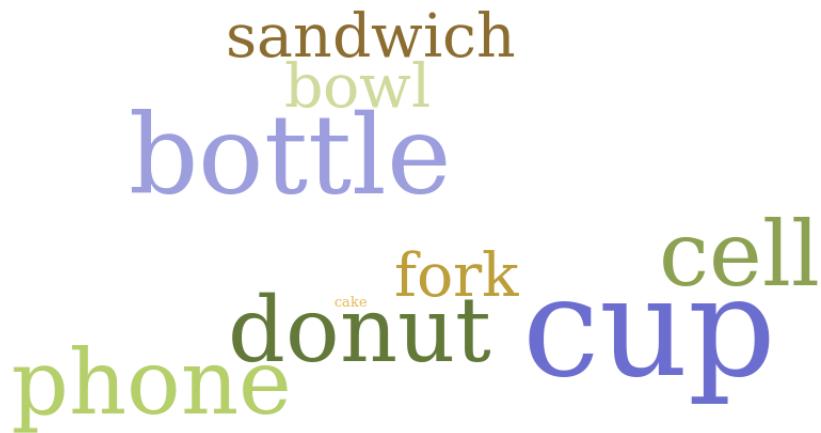


Figura 9: Wordcloud para as deteções da TinyYoloV3



Figura 10: Wordcloud para as deteções da YOLOv3

A ferramenta online utilizada para gerar as wordclouds está disponível no seguinte link : <https://www.jasondavies.com/wordcloud/>.

3.3 Análise das wordclouds obtidas

Observando as wordclouds geradas podemos concluir que todas as redes detetaram em maioria as mesmas keywords. Por exemplo, para todas as redes as keywords "cup", "sandwich", "bottle" distinguem-se pelo seu tamanho grande . Estas palavras são maiores que as outras palavras pois foram detetadas mais vezes nas imagens.

Algo que é possível concluir, pelas wordclouds geradas, é que a YOLOv3 é a rede mais coerente de todas, em que quase todas as mesmas palavras tem o mesmo tamanho. Por exemplo a retinaNet detetou "dog" e "pizza", no entanto essas deteções estão erradas, pois não estão presentes nas imagens, o que leva a uma grande discrepância no tamanho das palavras geradas

4 State-of-the-art

Foi dedicado bastante tempo na pesquisa de state-of-the-art e na sua implementação. Infelizmente, poucos resultados foram obtidos devido a falta de documentação, erros na compilação, alguma inexperiência e falta de tempo. No entanto, este assunto poderá ser novamente abordado de forma a serem feitas novas tentativas quando houver mais disponibilidade.

Apesar de poucos resultados terem sido obtidos, foi feita uma pesquisa de comparações entre as redes que se tentou implementar e a YOLOv3, de forma a pelo menos criar uma ideia do que esperar.

Após a pesquisa feita, considera-se que a YOLOv3 será a rede que produzirá melhores resultados.

4.1 Single Shot Detection

A informação do funcionamento desta rede encontra-se em [2]. Resumidamente, uma rede SSD funciona por tirar um unico shot e detetar multiplos objetos na imagem.

A rede discretiza o espaço de saída das caixas delimitadoras num conjunto de caixas padrão em diferentes proporções e escalas de aspecto por localização do mapa de features. No momento da previsão, a rede gera pontuações para a presença de cada categoria de objeto em cada caixa padrão e produz ajustes à caixa para corresponder melhor à forma do objeto. Além disso, a rede combina previsões de vários mapas de features com diferentes resoluções para lidar naturalmente com objetos de vários tamanhos

4.1.1 Resultados obtidos

A versão da rede SSD utilizada foi apenas treinada para detetar 20 keywords, é bastante precisa, no entanto devolve resultados inferiores à YOLOV3.

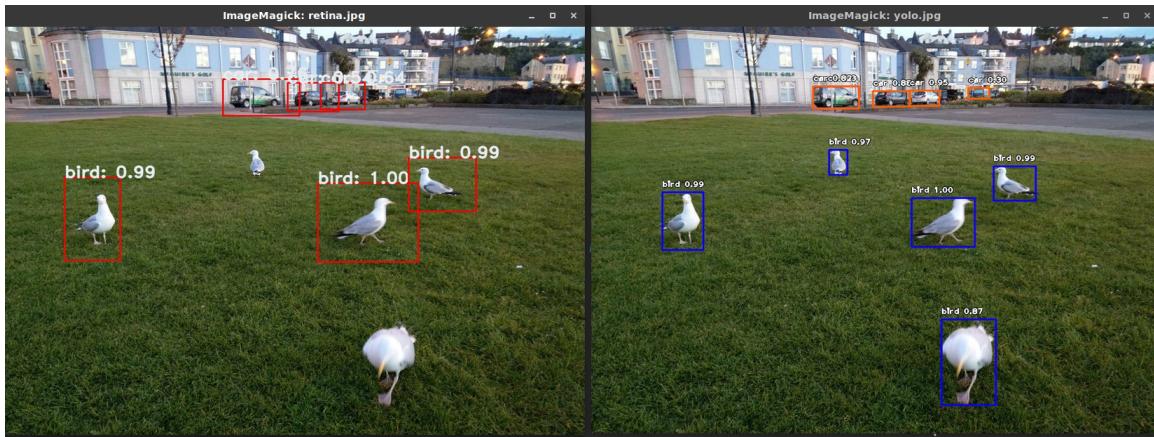


Figura 11: Comparação de deteções, entre a mesma imagem, para a rede SSD e para a rede YOLOv3

Analizando o teste demonstrado na figura 11 é fácil de verificar que a YOLO deteta mais "birds" que a rede SSD. Esta situação foi notável em mais testes realizados, pelo que se conclui que a rede YOLO é superior á rede SSD utilizada.

4.2 CornerNet Lite

No website da OpenCV foi descoberto o artigo [1] onde é explicado o funcionamento da rede CornerNet. Esta rede, em primeiro lugar, elimina o uso de caixas de ancoragem, que são populares nos métodos single-stage. A CornerNet representa uma caixa delimitadora como um par de pontos-chave, o canto superior esquerdo e o canto inferior direito. Segundo, o método introduz um novo tipo de pooling layer - corner pool - que ajuda a rede a localizar melhor os cantos. Por último, mas não menos importante, o desempenho! O Corner-Net alcança 42,2% AP no MS COCO, que é, como afirmam os autores, o melhor resultado em comparação com todos os detectores de uma etapa existentes na época.

Podemos ver uma comparação entre os resultados que os autores obtiveram e os resultados da YOLOv3 no dataset MSCOCO.

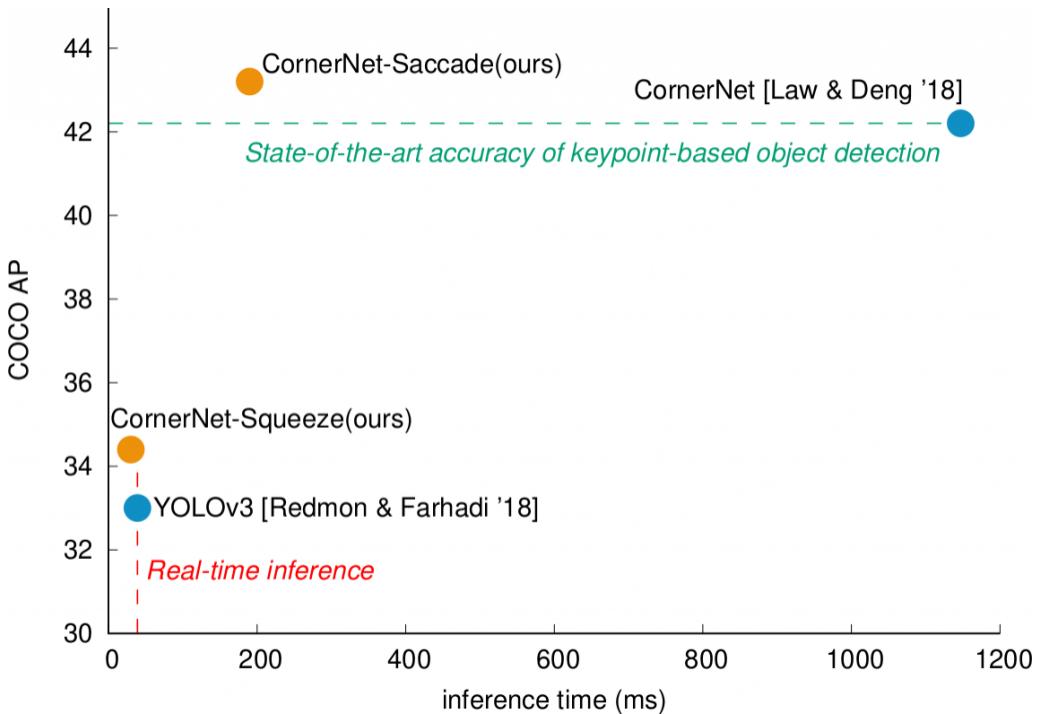


Figura 12: Resultados obtidos pelos autores.

4.2.1 Resultados obtidos

Infelizmente não foi possível obter qualquer tipo de resultados para esta rede, pois ocorreu um erro na compilação que até à data ainda não se conseguiu arranjar solução.

A rede utilizada vem de : <https://github.com/princeton-vl/CornerNet-Lite> e embora se tenha seguido a documentação á letra e terem sido realizadas no mínimo cerca de 15 tentativas diferentes não foi possível utilizar a rede.

Uma pesquisa nos comentários em <https://github.com/princeton-vl/CornerNet-Lite/issues> revelou duas coisas :

1. Mais utilizadores deparam-se com o mesmo erro, alguns conseguiram resolver mas outros não.
As soluções propostas nos comentários foram testadas mas não resultaram.
2. Alguns utilizadores afirmaram que a rede devolia, na realidade, piores resultados que a YOLOv3.

4.3 Cascade Mask R-CNN

Após alguma pesquisa de state-of-the-art o seguinte website foi encontrado : <https://paperswithcode.com/sota/object-detection-on-coco>. Este website funciona como repositório para comparação de state-of-the-art para diferentes redes/algoritmos (e os respetivos artigos) em diferentes áreas. O artigo que chamou á atenção foi o que se encontra nas referências em [3], visto que foi o que prometeu melhores resultados tal como visto na figura abaixo.

Object Detection on COCO test-dev

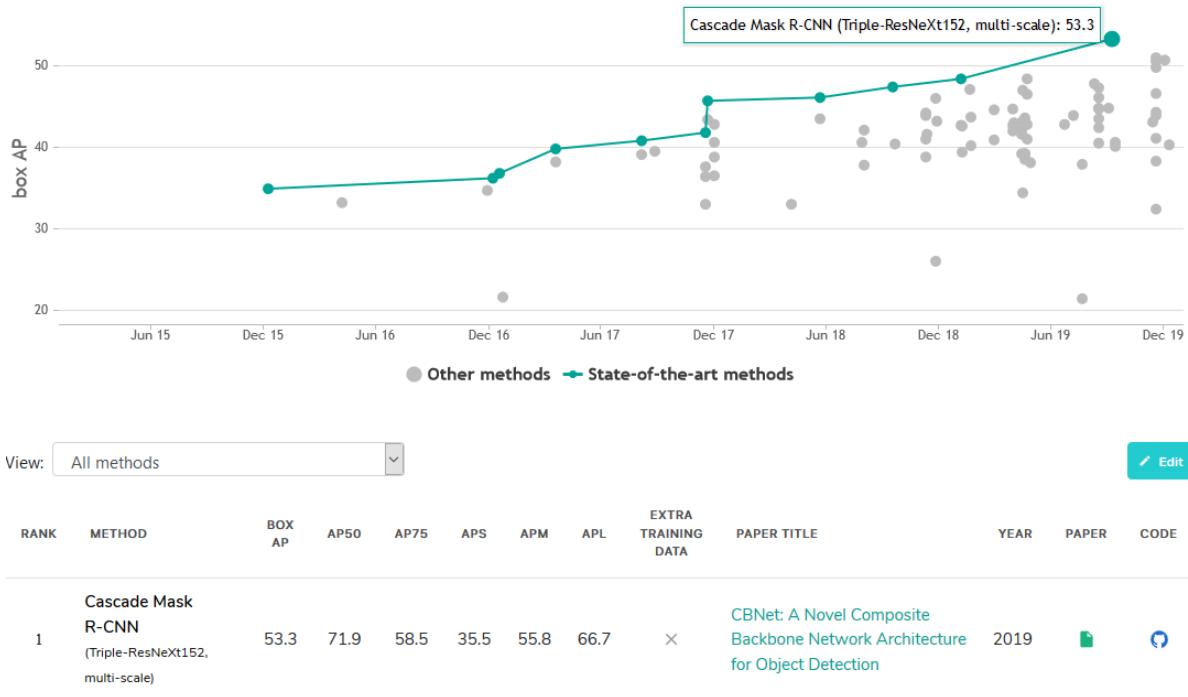


Figura 13: Resultados demonstrados no website.

Infelizmente havia pouca documentação em: <https://github.com/PKUbahuangliuhe/CBNet>. Juntando a falta de documentação aos constrangimentos de tempo e à inexperiência não foi possível testar o seu funcionamento.

4.4 Google Cloud Vision API

A Google Cloud Vision API, sugerida pelo professor, não foi colocada em funcionamento. Esta situação deveu-se a três motivos:

1. Falta de experiência a trabalhar com APIs.
2. Dificuldade na interpretação da documentação (passos a seguir para por a API a funcionar).
3. Constrangimentos de tempo.

Apesar de não haver resultados a apresentar, foi feita uma pesquisa de forma a comparar a rede YOLOv3 com a google cloud vision API. No seguinte link: <https://medium.com/sugarkubes/googles-object-detection-api-vs-yolov3-7956f9a9a6c5> é possível verificar comparações diretas entre YoloV3 e a Google Cloud Vision, onde a YOLOv3 obteve resultados bastante superiores. Sendo que esta é a única fonte encontrada de comparação direta, não há forma de comprovar a veracidade da afirmação.

Posto isto, assim que haja disponibilidade, será feita uma nova tentativa, mais aprofundada, de forma a colocar a API a funcionar e assim fazer as comparações necessárias com a YOLOv3.

5 Escrita do capítulo

Embora ainda esteja só em formato de "rascunho", relativamente à escrita do capítulo foram abordados e explicados os seguintes tópicos:

1. What is computer vision?
2. What is image classification?
3. What is object localization?
4. What is object detection?
5. What is object recognition?
6. What is image annotation?
7. What is image description?
8. What is a Dataset?
9. What is a Model?
10. What is a Category?
11. What is an object?
12. What Libraries are available?
 - (a) OpenCV.
 - (b) Tensorflow.
 - (c) VLFeat.
 - (d) BOOFCV.
 - (e) GluonCV.
13. Classification based algorithms.
14. Regression based algorithms.
15. What Deep Neural Networks and Algorithms are available? (Explicação de como funcionam)
 - (a) SqueezeNet.
 - (b) Resnet.
 - (c) InceptionV3.
 - (d) DenseNet.
 - (e) RetinaNet.
 - (f) YoloV3.
 - (g) Tiny YOLO V3.

6 Bibliografia

- [1] Review: SSD — Single Shot Detector (Object Detection)
<https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11>.
- [2] Latest Trends of Object Detection: From CornerNet to CenterNet Explained. Part I: CornerNet.
<https://opencv.org/latest-trends-of-object-detection-from-cornernet-to-centernet-explained-part-i-cornernet/>
- [3] Latest Trends in Object Detection: From CornerNet to CenterNet Explained. Part II: CornerNet-Lite
<https://opencv.org/latest-trends-in-object-detection-from-cornernet-to-centernet-explained-part-ii-cornernet-lite/>

Referências

- [1] Hei Law and Jia Deng. CornerNet: Detecting objects as paired keypoints, 2018.
- [2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.
- [3] Yudong Liu, Yongtao Wang, Siwei Wang, TingTing Liang, Qijie Zhao, Zhi Tang, and Haibin Ling. Cbnet: A novel composite backbone network architecture for object detection, 2019.