Detecção e Classificação de Objetos Presentes em Imagens Aéreas de Drones de Ambientes Urbanos

*Guilherme R. Sganderla, *Claudio Roberto M. Mauricio, †Valéria Nunes dos Santos e *Fabiana Frata F. Peres
*Universidade Estadual do Oeste do Paraná
Ciências da Computação - Centro de Engenharias e Ciências Exatas
Foz do Iguaçu, Paraná - Brasil
Emails: grodriguessganderla@gmail.com, crmmauricio@gmail.com, fabiana.peres@unioeste.br
†Fundação Parque Tecnológico Itaipu
Foz do Iguaçu, Paraná, Brasil
Email: valeria.valsnfw@gmail.com

***Resumo*—Através de grandes conjuntos de dados, é possível treinar e instruir uma máquina com habilidades para realizar tarefas antes realizadas somente por humanos. Essa possibilidade tem se tornado cada vez mais real com o uso de *Deep Learning* e algoritmos poderosos que vem sendo desenvolvidos ao longo do tempo. Entre eles está o YOLO, um algoritmo de Rede Neurais Convolucionais que permite diversos usos, entre eles a detecção e classificação de objetos contidos em imagens de ambientes urbanos, como pessoas e veículos, permitindo identificar e localizar os objetos dentro das imagens. Este trabalho apresenta um modelo para detecção e classificação de classes de objetos comuns em ambientes urbanos - Pessoas, Veículos-Pequenos, Veículos-Médios e Veículos-Grandes). Para esse projeto foi utilizado uma combinação de 3 *datasets* de imagens aéreas de drone de ambientes urbanos (*Stanford Drone Dataset*, *Vision Meets Drone*, *The Unmanned Aerial Vehicle Benchmark Object Detection and Tracking*). O resultado obtido do treinamento inicial desse algoritmo YOLO foi de uma acurácia média de 67,2%.**

***Abstract*—Through large data sets, it is possible to train and instruct a machine with skills to perform tasks previously performed only by humans. This possibility has become increasingly real with the use of Deep Learning and powerful algorithms that have been developed over time. Among them is YOLO, a Convolutional Neural Network algorithm that allows several uses, including the detection and classification of objects contained in images of urban environments, such as people and vehicles, allowing the identification and location of objects within the images. This work presents a model for detecting and classifying common object classes in urban environments - People, Small Vehicles, Medium-Vehicles and Large-Vehicles). For this project we used a combination of 3 datasets of aerial drone images of urban environments (Stanford Drone Dataset, Vision Meets Drone, The Unmanned Aerial Vehicle Benchmark Object Detection and Tracking). The result obtained from the initial training of this YOLO algorithm was an average accuracy of 67.2%.**

I. Introdução

Com a grande ascensão da capacidade computacional e da disponibilidade de grandes conjuntos de dados, a utilização de *Deep Learning* (DL) para a criação de sistemas inteligentes vem crescendo. Essa tecnologia permite capacitar uma máquina a compreender problemas que envolvem reconhecimento de padrões dentro de um conjunto de dados e possibilita uma máquina realizar certas tarefas que antes eram possíveis somente a humanos. Os sistemas inteligentes podem utilizar *hardwares*, como o *Unmanned Aerial Vehicles (UAV)* ou mais conhecido como drone para capturar imagens e dados de uma forma geral via sensores e câmeras.

A utilização de DL pode se estender a serviços públicos em cidades, se tornando um dos pilares do projeto e construção de uma *Smart City*. Uma *Smart City* prevê a utilização de tecnologias da informação e comunicação (TIC) [1] de ultima geração para desenvolvimento de um ecossistema controlado por sistemas conectados entre si.

O objetivo deste trabalho foi combinar as tecnologias de DL e UAV para detecção e classificação de objetos presentes em imagens aéreas de ambientes urbanos. As classes de objetos escolhidos para detecção e classificação foram:

- Pessoas: comum em ambientes urbanos e que para o contexto de *Smart City* pode fornecer recurso essencial para dezenas de aplicações no contexto de segurança, prevenções(aglomerações, lotação, entre outros), estatísticas(quantidade de pessoas em determinada área, turismo, etc.), e entre outros.
- Veículos-Pequenos: compreende de motos, bicicletas, entre outros. São comuns em áreas urbanas e podem ser útil a detecção para certas aplicações.
- Veículos-Médios: corresponde a carros, vans, caminhões pequenos, entre outros. É o objeto que tem mais presença na áreas urbanas e também um alvo de muitas aplicações.
- Veículos-Grandes: composto por caminhão, ônibus, entre outros. Tem uma presença pequena em áreas urbanas, mas devido ao seu tamanho ele pode ser motivo de diversos problemas (congestionamentos) e assim sua detecção pode ser útil.

Conforme um objeto é detectado, é apresentado o nome e a porcentagem de certeza da classe a qual o objeto pertence.

Para a implementação, foi utilizado o algoritmo de Rede Neural Convolucional (CNN) YOLO (*'You Only Look Once'*), uma família de arquitetura de modelos de detecção criado seguindo o padrão de um único estágio de detecção. Dessa forma permite obter um tempo de treinamento e inferência menor que outros modelos de detecção com baixa acurácia. A versão da YOLO utilizada é a v5, disponibilizada por Gleen

Jocher em maio de 2020, sendo uma versão não oficial da YOLO.

II. Conceitos Gerais

A maneira como um modelo de DL pode aprender depende do problema a qual ele pertence e a forma do aprendizado. Existem vários tipos de aprendizado a qual cada um está associado a forma como tratam os dados; dentre eles destacam-se:

- Aprendizado Supervisionado: Os dados que o algoritmo recebe são conjuntos de pares x e y, sendo a entrada e a saída, respectivamente. Este aprendizado consiste no algoritmo aprender o padrão que leva x resultar no y, para que ele possa responder o y de valores de entrada x nunca vistos antes. É definido como treinamento por exemplos e é utilizado principalmente em problemas de classificação e regressão.
- Aprendizado Não Supervisionado: Os dados que o algoritmo recebe são conjuntos de entradas, comumente chamadas de x. O algoritmo tenta identificar similaridades entre as entradas para descobrir atributos comuns e formar grupos de dados para representar os atributos que os dados tem em comum; são utilizados em problemas do tipo *clustering*.
- Aprendizado por Reforço: Não é fornecido nenhum dado de entrada ao algoritmo. O algoritmo tem dois atores, um agente e o ambiente e conforme cada interação é realizada pelo agente com o ambiente é atribuído uma recompensa e uma punição. O objetivo do algoritmo é fazer o agente maximizar a recompensa das ações feitas.

A maioria dos problemas podem ser minimizados em um modelo, sendo os mais conhecidos:

- Classificação: através de dados de entrada o algoritmo retorna um valor representando a classe do objeto que o dado de entrada representa. A classificação pode ser binária ou de multi-classes. Na classificação binária, um objeto é classificado como pertencente ou não a uma classe; ou ainda o objeto é classificado entre dois possíveis tipos de objetos. Já na classificação de multi-classes o resultado da predição de uma entrada resulta da análise dos valores obtidos. Cada classe é representada por um intervalo de valores [2].
- Regressão: através dos dados de entrada o modelo tenta criar uma função que aproxima a função real utilizada para resultar os dados exemplos; por exemplo, a partir de dados resultantes de uma função tipo Teorema de Pitágoras, ele tenta criar uma função parecida. Esse algoritmo lida com números reais e prevê dados numéricos futuros reais. [2].
- *Clustering*: através dos dados analisados, busca agrupa-los por similaridades e desse modo definir classes. Nesse modelo não é necessário conhecer previamente as classes dos dados. O modelo define as classes e agrupa os dados. Quando um novo dado é fornecido, ele o classifica pelo atributo em comum [2].

III. Materiais e Métodos

Para a implementação do sistema de detecção e classificação foi utilizado uma máquina contendo uma placa de vídeo RTX 2080Ti com suporte ao CUDA 11, e mais 32GB de memória RAM, favorecendo a melhora do desempenho na etapa de treinamento.

O conjunto de imagens aéreas de ambientes urbanos utilizados no treinamento, na validação e nos testes resultou da combinação de 3 (*datasets*):

- *Stanford Drone Dataset*(SDD): Um *dataset* criado pelo *Computacional Vision and Geometry Lab* da *Stanford*, que possui vídeos de diversas cenas e lugares do campus da universidade de *Stanford*. Cada vídeo relata o cotidiano do campus, onde estão presentes pedestres, veículos de locomoção pequenos (bicicletas e carrinhos), carros, e ônibus. Sendo a maior concentração do *dataset* os pedestres e ciclistas [3].
- *Vision Meets Drone*(VisDrone): Um *dataset* criado pelo time AISKYEYE no *Lab of Machine Learning and Data Mining* na *Tianjin University* da China. Ele apresenta diversas imagens em áreas urbanas de 14 cidades diferentes em tempo e condições diferentes [4].
- *The Unmanned Aerial Vehicle Benchmark Object Detection and Tracking*(UAVDT): Um *dataset* focado em detecção de veículos em diversas áreas com diversos climas e horários [5].

É importante ressaltar que cada *dataset* apresenta uma forma própria de anotação para os objetos encontrados nas suas imagens. Para adequar e combinar os 3 *dataset*, foi necessário padronizar a forma das anotações dos objetos de cada imagem. A forma utilizada foi converter todas as anotações dos *dataset* para o formato utilizado pela YOLOv5.

A conversão utiliza o valor médio da caixa delimitadora para os eixos x e y, sendo exemplificado na figura 1. Para a conversão foi implementado um *script* na linguagem *Python*. O script percorre as anotações dos 3 *datasets*, extrai os limites das caixas delimitadoras e calcula os valores médios de cada objeto anotado e guarda esses valores em um outro arquivo. Esse processo se repete para cada anotação dentro do conjunto.

A combinação desses 3 *datasets*, feito algumas otimizações e resolução de problemas, resultou em 187.138 imagens. Foi usada a proporção de 70%, 15% e 15% para treino, teste e validação respectivamente. A quantidade de imagens utilizadas de cada *dataset* em cada etapa é apresentado na tabela I.

Quanto aos objetos anotados, a combinação dos *datasets* possuem 4.112.482 objetos anotados, sendo a divisão deles apresentado na tabela II.

Para realizar o treinamento e a inferência optou-se por utilizar o algoritmo YOLOv5 com a biblioteca de *Machine Learning PyTorch*. A escolha da YOLOv5 como algoritmo de detecção e classificação de objetos foi feita por 3 motivos:

- Simplicidade de uso: Através do *"clone"* do repositório principal da YOLOv5, é possível instalar e rodar sessões de treinamentos sem muito esforço. Além de treino, ela

Figure 1. Exemplo de uma anotação YOLOv5 [6]

Table I
TABELA DE PROPORÇÃO DAS IMAGENS

Dataset	Treino	Teste	Validação
SDD	73.092	15.663	15.663
VisDrone	29.686	6.301	6.324
UAVDT	28.341	5.970	6.098
Total	131.119	27.934	28.085

possui modelos pré-treinados para uso, e para servir também como pesos iniciais para o treino de um novo modelo.

- Velocidade de treinamento: Comparado com os modelos de detecção de 2 estágios, a YOLOv5 possui uma velocidade de treinamento maior ao custo de uma acurácia menor. É disponibilizado pela YOLOv5 diversos modelos, onde a diferença de cada um está na quantidade necessária de tempo de treino e na acurácia média resultante, sendo o menor modelo o mais rápido e menos preciso e o maior modelo o mais demorado e mais preciso.
- Velocidade de inferência: A taxa de *Frames per second* (FPS) do resultado da inferência de um modelo YOLOv5 qualquer pode atingir e ultrapassar 60 FPS, assim apresentando um resultado rápido ao usuário. Esse ponto é decisivo quando a aplicação precisa de uma inferência em tempo real, que no nosso caso, é necessário [6].

Table II
TABELA DE PROPORÇÃO DAS ANOTAÇÕES

Classe	Treino	Teste	Validação	Total
Pessoas	1.039.328	225.216	220.918	1.485.462
Veículos-Pequenos	460.431	100.167	98.148	658.746
Veículos-Médios	1.287.414	272.524	274.308	1.834.246
Veículos-Grandes	94.052	20.091	19.885	134.028

Table III
TABELA DO RESULTADO DO TESTE

Classe	P	R	mAP@0.5	mAP@0.5:0.95
all	0.752	0.628	0.672	0.358
pessoa	0.682	0.609	0.626	0.225
veiculo-pequeno	0.737	0.65	0.669	0.251
veiculo-medio	0.856	0.897	0.916	0.597
veiculo-grande	0.843	0.837	0.877	0.612

A escolha da utilização da biblioteca *PyTorch* foi devido a implementação oficial da YOLOv5 ser nessa biblioteca. O modelo utilizado no treinamento do *dataset* foi o YOLOv5s, sendo a versão pequena da YOLOv5, com menos parâmetros para treinar e menos tempo exigido de treino.

IV. EXPERIMENTOS

Foi feito um treinamento do modelo YOLOv5s com o seguintes parâmetros:

- *batch-size*: foi utilizado 55 de *batch* devido a alta capacidade de memória da máquina de treino. Isso significa que a cada etapa do treinamento será utilizado um conjunto de 55 imagens, agilizando o processo.
- *epochs*: as épocas são quantidade de vezes que o treino será iterado sobre o conjunto total de imagens; quanto maior o número de épocas, maior será a quantidade que o modelo irá treinar sobre o conjunto de imagens. Se o número de épocas for muito grande, pode ocorrer o *Overfitting*, que corresponde ao modelo perder a capacidade de generalizar resultando na memorização das imagens de entrada. Para esse parâmetro foi utilizado o valor 50 para ser um valor de teste, que subsequentemente será testado com outros valores.

Com os parâmetros definidos, foi realizado um treinamento considerando o conjunto de imagens, sendo cada época de treinamento terminada em pelo menos 1 hora. Após 50 épocas, o modelo passou por uma avaliação para determinar a acuraria e outras métricas, o resultado do teste são apresentado na tabela III:

Na tabela III a primeira linha com a classe *all* representa o total do modelo e cada linha subsequente representa a classe dos objetos. A coluna P representa a *Precision*, que avalia a relevância dos objetos recuperados. A coluna R representa o *Recall*, que evidencia se os objetos importantes foram recuperados. As ultimas duas colunas são as porcentagens a partir da *Mean Average Precision* (mAP), que indica o desempenho médio do modelo e é a métrica que determina a qualidade do modelo. Além do calculo de *Precision, Recall* e mAP, o teste ou validação do modelo YOLOv5 fornece diversas outras métricas e a matriz de confusão (apresentada na tabela IV).

A tabela IV apresenta o resultado do teste comparando com o resultado verdadeiro. Cada linha representa a classe prevista e a coluna representa a classe verdadeira. Cada célula

Table IV
TABELA DO RESULTADO DO TESTE

	pessoa	veiculo pequeno	veiculo médio	veiculo grande	outros	back. FP
pessoa	0.67	0.05			0.01	0.43
veiculo pequeno	0.02	0.68				0.13
veiculo medio		0.01	0.91	0.08		0.28
veiculo grande				0.81		0.02
outros		0.01			0.21	0.13
back. FN	0.31	0.25	0.08	0.10	0.78	

contêm o valor da porcentagem onde a classe prevista é igual a classe verdadeira. Duas classes foram adicionadas, a *background FP e FN*, essas classes são exclusivas do testes e representam os valores de Falso Positivo(alarme falso) e Falso Negativo(rejeção correta).

Analisando a tabela IV é possível notar os problemas encontrado nas duas classes pessoa e veículo-pequeno. Ambas são compostas de objetos pequenos e tem os menores valores de mAP. Através das combinações das linhas e colunas de ambas classes, observa-se que elas acabam sendo confundidas, uma com a outra.

Outra forma de analisar os dados da tabela III é através das figuras 2 e 3, sendo a métrica mAP o resultado da combinação das duas métricas em um calculo, sendo apresentado o resultado na figura 4

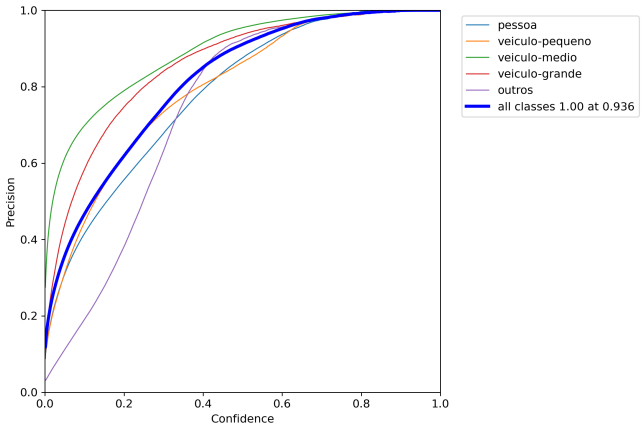

Figure 2. Gráfico referente a métrica *Precision* com a confiança

A métrica de precisão ou *Precision* é calculada através do:

$$\frac{VerdadeiroPositivo}{VerdadeiroPositivo + FalsoPositivo}$$

Onde o Verdadeiro Positivo é quanto a predição está correta (a classe foi dita como Pessoa e era verdadeira) e Falso Positivo que são os erros que o modelo fez durante a predição, onde ele prediz a classe como pessoa mas no caso não é. Essa métrica calcula a qualidade do modelo em evitar falsos alarmes [7].

A figura 2 apresenta o aumento da precisão do modelo com base nas taxas de confianças, que é o valor da porcentagem de intersecção da caixa delimitadora prevista com a caixa delimitadora real.

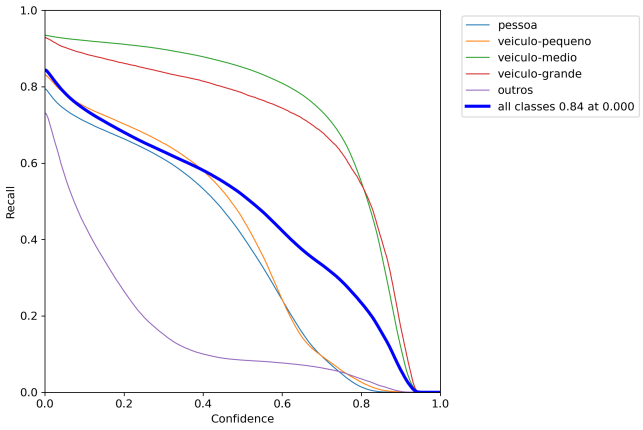

Figure 3. Gráfico referente a métrica *Recall* com a confiança

Para a métrica de *Recall*, o calculo é feito através da formula:

$$\frac{VerdadeiroPositivo}{VerdadeiroPositivo + FalsoNegativo}$$

Onde nesse caso o VP se mantém o mesmo da anterior e agora com a adição do Falso Negativo, é quando a predição é dita como errada e isso é falso (a classe prevista foi dita que não era Pessoa e era).

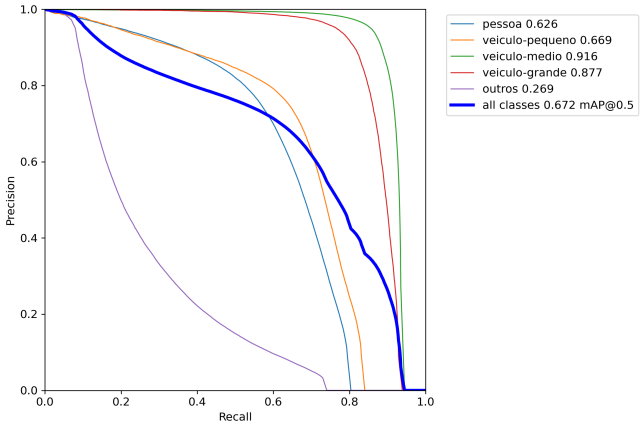

Figure 4. Gráfico referente a métrica *Precision* com a *Recall* formando a mAP

Através da figura 4, é possível ver a métrica mAP com base na *Precision* sobre a *Recall*. Ela é obtida fazendo o calculo da Precisão Média [8] e utilizando a precisão média de cada classe e fazendo a média delas [9]:

$$AvegP = \int_0^1 p(r)\,dr$$

$$mAP = \frac{\sum_{q=1}^{Q} AvegP(q)}{Q}$$

Onde Q é o número de classes.

Com a realização do teste, também é possível obter imagens com resultados de detecção, para demonstrar como é apresentado o resultado da inferência das imagens. Esse resultado é apresentado na figura 5 e tem presente uma imagem e as detecções feitas.

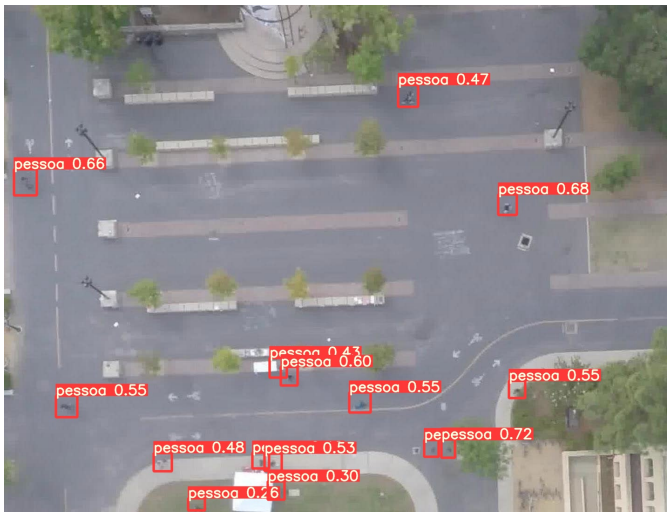

Figure 5. Imagem resultante de uma detecção

V. Conclusão

Através dos resultados obtidos com o experimento realizado observou-se que a utilização de imagens áreas de drones para a detecção e classificação de objetos apresenta resultados promissores; desse modo a proposta de utilizar Drones com *Deep Learning* fornece informações relevantes para o contexto de *Smart Cities*. Além disso este trabalho demonstra o estado da arte da tecnologia *Deep Learning*, sendo o foco nesse projeto o algoritmo YOLOv5. O algoritmo possibilitou um treinamento e uma inferência rápida para a detecção de objetos e com uma acurácia razoavelmente boa para algumas classes de objetos.

O resultado do treinamento possibilitou comprovar que a combinação do 3 *datasets* (SDD, VisDrone e UAVDT) foi adequado, sendo encontrado um pequeno problema com a classe pessoa. A classe pessoa é a classe com o segundo maior número de objetos anotados, mas é a classe com o pior mAP entre as 4 classes. Sendo o motivo dessa baixa mAP desconhecido; especula-se que possa ser que não tenha uma diferença entre o objeto anotado e o ambiente onde ele se encontra.

Desconsiderando o problema com a classe pessoa, foi apresentado resultados razoavelmente bom para uma experiência inicial com a combinação de tecnologias e *datasets*. A possibilidade de melhorar o desempenho de um modelo treinado é possível na YOLOv5, sendo necessário a realização de um novo treinamento com os parâmetros do modelo aprimorados.

Com os resultados obtidos, pretende-se como próximos passos realizar um novo treinamento para tentar atingir um desempenho maior.

References

[1] K. Su, J. Li, and H. Fu, "Smart city and the applications," in *2011 international conference on electronics, communications and control (ICECC)*. IEEE, 2011, pp. 1028–1031.

[2] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2012. [Online]. Available: https://books.google.com.br/books?id=maz6AQAAQBAJ

[3] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *European conference on computer vision*. Springer, 2016, pp. 549–565.

[4] P. Zhu, L. Wen, D. Du, X. Bian, Q. Hu, and H. Ling, "Vision meets drones: Past, present and future," *arXiv preprint arXiv:2001.06303*, 2020.

[5] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian, "The unmanned aerial vehicle benchmark: Object detection and tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 370–386.

[6] G. Jocher, A. Stoken, J. Borovec, NanoCode012, A. Chaurasia, TaoXie, L. Changyu, A. V, Laughing, tkianai, yxNONG, A. Hogan, lorenzomammana, AlexWang1900, J. Hajek, L. Diaconu, Marc, Y. Kwon, oleg, wanghaoyang0106, Y. Defretin, A. Lohia, ml5ah, B. Milanko, B. Fineran, D. Khromov, D. Yiwei, Doug, Durgesh, and F. Ingham, "ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations," Apr. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4679653

[7] S. Skansi, *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*, ser. Undergraduate Topics in Computer Science. Springer International Publishing, 2018. [Online]. Available: https://books.google.com.br/books?id=5cNKDwAAQBAJ

[8] M. Zhu, "Recall, precision and average precision," 09 2004.

[9] C. D. Manning and P. Raghavan, "and schutze, h.[2008] introduction to information retrieval," 2008.