

Detecção e reconhecimento de animais em vídeos de segurança utilizando técnicas de processamento de imagens

Adriano Marques Martins¹, Antônio Carlos Fava de Barros¹

¹ Universidade Federal de Viçosa – Campus Florestal
Rodovia LMG 818, km 06 – 35690-000 – Florestal – MG – Brasil

{adriano.martins;antonio.fava}@ufv.br

Resumo. A expansão imobiliária provoca a ocupação de regiões próximas às áreas rurais, onde é comum a presença de animais próximos à presença do homem. Estes animais podem perturbar a ordem, causando transtornos, danos ou prejuízos, e até mesmo colocando em risco a segurança das pessoas. Neste projeto, foi desenvolvido um modelo de visão computacional com o objetivo de identificar a presença destes animais em imagens de vídeos de segurança, seja para simples controle de presença, monitoramento e até mesmo alarme de perigo. O modelo foi desenvolvido usando o conjunto de imagens obtidas do Open Images Dataset e treinado através do framework Darknet, obtendo resultados validados pela métrica mAP (mean Average Precision) com precisão media geral de 84.58% em prever e detectar os objetos das seguintes classes: Cavalo, Pessoa, Gado, Porco, Gato, Cachorro, Onça e Cobra.

1. Introdução

O aprendizado de máquina (*machine learning*) é uma das áreas da inteligência artificial onde um sistema tem a capacidade de aprender através de padrões em dados, que podem ser de diversos tipos, desde números à imagens. Nos últimos anos, o aprendizado de máquina está sendo empregado em diferentes aplicações, tais como sistemas de recomendações de filmes, filtragem de conteúdos em redes sociais, transcrição de textos, dentre outras [LeCun et al. 2015].

Já a visão computacional (*computer vision*) é uma área que auxilia o computador a obter dados de uma imagem digital, extraíndo características que permite, por exemplo, identificar objetos. Com a popularização do aprendizado profundo (*deep learning*), a visão computacional cada vez mais vai se associando a estas tecnologias, ganhando destaque com aplicações em diferentes áreas como saúde, esportes, robótica e carros autônomos [Karn 2021].

Dentre os algoritmos de aprendizado profundo, as redes neurais convolucionais (ConvNet/CNN) são algoritmos que recebem dados na sua entrada, como uma imagem digital, e atribui diferentes pesos para suas camadas internas durante o treinamento, de acordo com as características presentes na imagem. Ao final de uma etapa de treinamento, o algoritmo deve ser capaz de classificar as imagens obtidas em classes definidas. Diversas redes pré-construídas de detecção de objetos em imagem estão sendo amplamente empregadas dentro da visão computacional como o CNN (R-CNN), Fast R-CNN, Faster R-CNN e o YOLO [Garg et al. 2018, Turečková et al. 2020, LeCun et al. 2015]. Estas redes apresentam excelente desempenho em problemas relacionados à visão computacional [Garg et al. 2018].

No campus da Universidade Federal de Viçosa, na cidade de Florestal/MG, é comum a presença de animais transitando nas áreas comuns. Estas situações podem provocar transtornos, como a transmissão de doenças, destruição do patrimônio e até mesmo situações de risco à vida humana.

Este trabalho propõe um modelo de visão computacional para a detecção e reconhecimento de animais que são comuns na região, como cachorros, gatos, cavalos, porcos, gados, e até mesmo cobras e felinos silvestres, como onças, visando garantir um controle de presença para o monitoramento e segurança.

As etapas da construção do modelo consistem na aquisição de imagens da Open Images Dataset [Kuznetsova et al. 2020] e na implementação de uma arquitetura YOLO (*You Only Look Once*) utilizando o framework Darknet [Redmon 2016].

As seções deste trabalho se dividem em: trabalhos relacionados (seção 2); uma breve descrição sobre o YOLO (seção 3); a metodologia utilizada, com a aquisição do banco de imagens, o treinamento da rede e a métrica de avaliação do modelo (seção 4); a apresentação dos resultados obtidos (seção 5); e por fim, a conclusão (seção 6) e os trabalhos futuros (seção 7).

2. Trabalhos Relacionados

Uma rede neural convolucional foi usada por [Schneider et al. 2018] para detectar objetos de interesse capturados por armadilhas fotográficas utilizadas para controle e monitoramento da vida ecológica. Os autores usaram as redes faster R-CNN e YOLO na segunda versão para detectar, classificar e quantificar espécies de animais. As imagens usadas estavam distribuídas em dois bancos de imagens, sendo um destes, rotulados manualmente. Os bancos foram agregados em um único conjunto de treinamento, apresentando melhores resultados finais com o modelo faster R-CNN, enquanto o YOLO falhou em desempenho.

Em [Turečková et al. 2020] foi utilizada uma rede YOLO na versão *tiny* para a detecção de faces de cachorros em imagens, distribuídas em dois bancos de imagens combinados para o treinamento do modelo. O modelo apresentou resultados com 0.92 de precisão média para um IoU (*Intersection over Union*) de 0.5, com uma velocidade de 0.012s para detectar a face de um cachorro em uma imagem de tamanho de 300×300 pixels usando um dispositivo móvel.

Enquanto em [Moallem et al. 2021], os autores usaram uma rede convolucional em dois estágios, sendo o primeiro estágio para detectar a presença de pássaros em um parque no Texas, e a segunda para a classificação destes pássaros. Os autores obtiveram uma precisão para a detecção com sensibilidade superior a 93%, uma especificidade de 92% e taxa média de IoU (*Intersection over Union*) de 68%. Para o sistema de classificação obtiveram sensibilidade geral de 93% e especificidade de 96%.

Como podemos observar, as redes neurais convolucionais, incluindo o YOLO, tem apresentado bons resultados na detecção e classificação de objetos, incluindo os animais.

3. YOLO

O YOLO (*You Only Look Once*) [Redmon et al. 2016] é uma arquitetura para detecção de objetos que utiliza uma rede neural profunda. A detecção de objetos é uma tarefa que consiste em localizar e classificar objetos que estão presentes em uma imagem.

A família de redes R-CNN se utilizam de diferentes regiões para localizar objetos, ou seja, a rede não analisa a imagem como um todo, e sim apenas as partes de interesse [Girshick 2015]. Por sua vez, o YOLO, utiliza uma única instância da imagem para detectar os objetos, trazendo diversas vantagens para a sua velocidade.

O YOLO também é rápido quando se trata de treinar o modelo, pois a arquitetura necessita que todas as imagens de treinamento e validação tenham arquivos de anotações contendo a região e a classe dos objetos presentes. Desta forma, durante o treinamento, o sistema

poderá identificar nas imagens apenas a região especificada, facilitando assim o aprendizado do modelo.

O modelo receberá uma imagem como entrada, que será então dividida em uma grade quadrangular de $S \times S$ células. Cada uma destas células será responsável por prever B caixas delimitadoras que conterão uma pontuação de confiança com a certeza daquela caixa conter um objeto. Estas caixas são compostas pelas coordenadas do centro da caixa (x, y) prevista, a altura h e largura w , além do valor de confiança na detecção do objeto. Assim, cada célula será responsável por prever uma única classe do objeto como pode ser visto na figura 1.

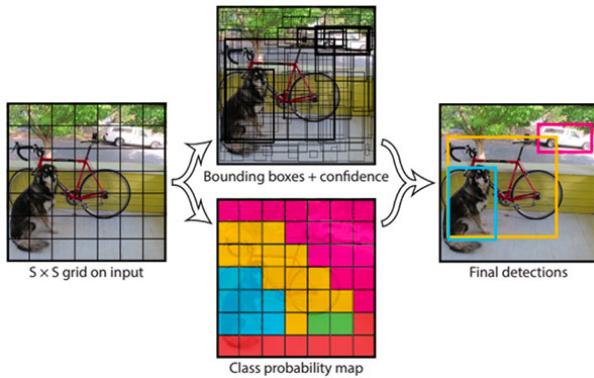


Figura 1. Divisão em uma grade, caixas delimitadoras por célula, o mapa de probabilidade por classe e as caixas delimitadoras resultantes [Redmon et al. 2016].

Entretanto, como pode ser visto na figura 1, boa parte destas caixas poderão ter valores muito baixos quanto a sua confiança. Para evitar este problema, pode-se definir um limiar de corte (*threshold*) que irá remover as caixas que não superarem este limite, impedindo que detecções pouco confiantes sejam consideradas.

Outro parâmetro que pode ser considerado é o valor da supressão não-máxima (*Non-Max Suppression*) que irá suprimir todas as caixas delimitadoras que possuem uma área compartilhada, ou seja, caixas que muito provavelmente estão identificando um mesmo objeto.

3.1. Arquitetura

Na sua terceira versão, o YOLO, apesar de não ser mais rápido que a sua versão anterior, apresentou uma melhor eficiência em suas predições, pois é capaz trabalhar com a imagem em 3 diferentes escalas, reduzindo a imagem 32, 16 e 8 vezes, para assim conseguir manter a acurácia mesmo em tamanhos menores da imagem [Redmon and Farhadi 2018].

O YOLOv3 tem uma arquitetura variante da darknet original denominada darknet-53. A arquitetura conta com 106 camadas convolucionais, sendo 53 camadas para a extração de características e outras 53 camadas para realizar a tarefa de detecção, como pode ser vista na figura 2. A YOLOv2 contava com apenas 30 camadas convolucionais [Redmon and Farhadi 2017], por isso a terceira versão apresenta uma redução na velocidade em troca de melhores resultados.

Nosso trabalho foi desenvolvido utilizando a YOLO na quarta versão, que apesar de manter o mesmo *backbone* da versão 3, apresenta uma performance ainda mais rápida que sua sucessora, devido a implementação de alguns recursos adicionais [Bochkovskiy et al. 2020].

4. Metodologia

Esta seção descreve a metodologia proposta com o conjunto de dados utilizado, resume o processo de treinamento e especifica as métricas usadas para avaliar o modelo. Na

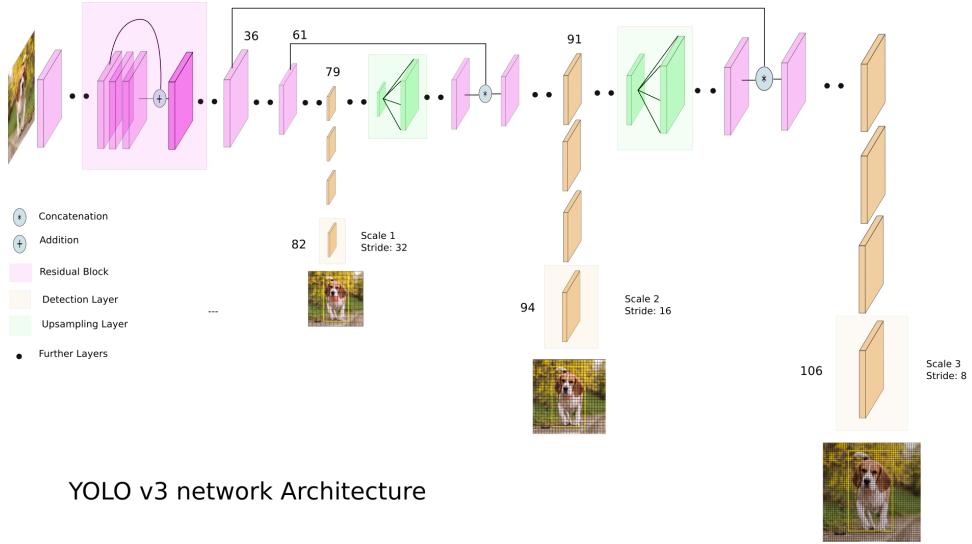


Figura 2. Arquitetura do YOLOv3 [Kathuria 2018].

seção seguinte serão apresentados os resultados obtidos.

4.1. Banco de Imagens

O conjunto de imagens de treinamento foi obtidos através do banco de imagens online denominado Open Images dataset v6 [Krasin et al. 2017] usando a ferramenta OIDv4 ToolKit¹. Inicialmente, o *dataset* foi inicializado com 500 imagens para cada uma das sete classes proposta para a criação deste modelo, totalizando 3500 imagens. As sete classes usadas no treinamento do modelo são: Cavalo, Pessoa, Gado, Porco, Gato, Cachorro, Onça e Cobra. Já o conjunto de validação foi formado por 200 imagens para cada uma das sete classes, totalizando 1400 imagens para validação.

4.1.1. Anotações

Após criação do banco de imagens, foi possível visualizar as imagens e seus respectivos arquivos de anotação contendo as âncoras de cada um dos objetos presentes. Cada linha desses arquivos contém os dados necessários para identificar aquele objeto durante a fase de treinamento e a fase de validação do modelo, sendo que o primeiro valor o nome da classe do objeto seguido pela coordenada da caixa que o delimita [Kuznetsova et al. 2020].

Entretanto, o formato de anotação da ferramenta OIDv4 é disponibilizado no formato PASCAL_VOC [*class_name*, x_{min} , y_{min} , x_{max} , y_{max}] que forma um retângulo através de dois pontos, enquanto o formato YOLO utiliza o formato [*class_id*, x_{center} , y_{center} , *width*, *height*], onde o ponto (x_{center} , y_{center}) corresponde ao centro da imagem e os valores *width* e *height* correspondem a largura e a altura, respectivamente, da caixa delimitadora.

Outro detalhe importante é que os valores que identificam a caixa delimitadora, no formato YOLO, devem estar normalizados entre 0.0 e 1.0. Desta forma, fez-se necessária a conversão dos arquivos de anotação adquiridos para o padrão aceito pelo YOLO.

¹https://github.com/EscVM/OIDv4_ToolKit

4.1.2. Seleção das imagens

A seleção das imagens, tanto de teste quanto de treinamento, é uma etapa importante quando se trata de aprendizado de máquina, pois imagens com pouca representatividade da classe podem levar o modelo a um aprendizado errado ou enviesado.

Para evitar estes possíveis problemas com o modelo, as imagens foram analisadas de forma bem criteriosa, e aquelas que foram identificadas com baixa representatividade da classe selecionada foram removidas do banco de imagens. Dentre os problemas encontrados nessas imagens e nos arquivos de anotações, os principais foram:

- Um mesmo objeto ter várias caixas anotados ao invés de uma única
- Objetos pouco representativos para o contexto
- Imagens que continham mais objetos que os considerados no arquivo de anotação
- Imagens contendo apenas partes dos animais, sem apresentar o todo.

Na figura 3, é apresentado um mosaico de algumas imagens que foram removidas manualmente afim que melhorar a acurácia do modelo.



Figura 3. Exemplo de imagens removidas antes do treinamento.

Em contrapartida, na figura 4, são apresentados alguns exemplos de imagens que foram aproveitadas, seja no treinamento ou na validação do modelo.

Como algumas imagens utilizadas possuem mais de uma única classe ou objeto, como podemos ver na figura 4, não faria sentido quantificar a quantidade de imagens e sim as instâncias de classes por imagem de acordo com a tabela 1.

4.2. Treinamento

O ambiente de desenvolvimento utilizado foi o Google Colab¹, sendo usado o framework Darknet [Redmon 2016] para simplificar e agilizar o processo de treinamento do modelo, pois é possível compilar o framework para utilizar a GPU integrada do Google Colab.

¹<https://colab.research.google.com/>



Figura 4. Exemplo de imagens que foram aproveitadas.

Tabela 1. Tabela com o número de instâncias por classe.

Classe	Teste	Treino	Total
Cavalo	185	825	1010
Pessoa	532	2770	3302
Gado	201	1857	2058
Porco	166	828	994
Gato	212	604	816
Cachorro	229	703	932
Onça	86	485	571
Cobra	180	503	683
Total	1791	8575	10366

Durante o treinamento foi utilizado a técnica de transferência de aprendizagem (*transfer learning*). Como a rede convolucional é formada por camadas de extração de características e reconhecimento de objetos [Hussain et al. 2019, Tan et al. 2018], podemos aproveitar os pesos originais da arquitetura, e retreinar a rede somente para obter os pesos para as camadas de reconhecimento. Utilizando os pesos originais para inicializar o modelo, a rede foi treinada por 8700 *epochs* com as classes dispostas na tabela 1.

4.3. Métricas

Uma métrica importante para poder medir o desempenho dos modelos de uma rede neural convolucional na detecção de objetos é o mAP (*mean Average Precision*) [Adami et al. 2021]. Quando um objeto é detectado e classificado, o algoritmo irá retornar a caixa delimitadora que delimita o objeto. Para que a classificação seja validada, é realizada a verificação das coordenadas do objeto detectado com os dados reais do mesmo contidos no respectivo arquivo de anotação.

Depois, é calculado o IoU (*Intersection over Union*) que nada mais é que a porcentagem de sobreposição entre a caixa detectada e a caixa real, onde ocorre 0 quando duas caixas

não estão sobrepostas e 1 quanto as caixas são idênticas, como pode ser visto na figura 5. Dado um limiar de aceite, podemos definir se aquele objeto detectado é semelhante ao objeto real identificado.

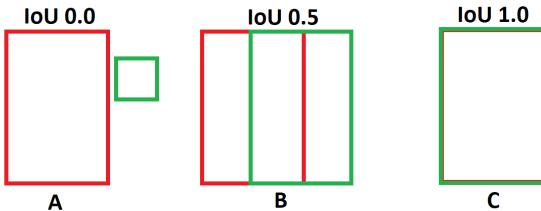


Figura 5. Caixas delimitadora predita em vermelho e a real em verde

$$IoU = \frac{\text{área de interseção}}{\text{área de união}}$$

Desta forma, o IoU nos permite identificar um verdadeiro positivo (TP) ou um falso positivo (FP). Dada essas medidas podemos encontrar a precisão média (*Average Precision*) que nos indica a porcentagem de acertos encontrados.

O mAP por sua vez corresponde a precisão média considerando todos os erros e acerto de todas as classes, permitindo ter uma visão geral do desempenho do modelo.

A função mAP, disponibilizada pelo framework Darknet, informa os seguintes valores: o número de instâncias previstas corretamente (TP), o número de instâncias previstas incorretamente (FP) e o número de instâncias que deixaram de ser previstas (FN). Através destas quantificações, a função reporta a acurácia, a revogação, a precisão e o F1-score como medida de desempenho para o nosso modelo.

5. Resultados

Utilizando 1791 instâncias das classes de teste, o modelo obteve uma média geral de 84.58% de acurácia (mAP) para um IoU de 0.5 e um limiar de corte (*threshold*) de 0.25 de certeza da classe obtida. Os resultados, por classe, podem ser observados na tabela 2.

Tabela 2. Resultado da função mAP do Darknet.

Classe	AP(%)	TP	FP
Cavalo	78.75	149	34
Pessoa	48.34	211	83
Gado	82.80	170	60
Porco	91.74	130	4
Gato	93.80	195	20
Cachorro	93.05	208	23
Onca	91.76	69	2
Cobra	96.39	156	4

Outras métricas resultantes foram a precisão do modelo de 85%, a revogação de 72%, o F1-score de 78% e a média de precisão da caixa delimitadora, o IoU, de 69.06%. Foram detectadas um total de 4865 objetos, contra apenas 1971 instâncias reais anotadas.

A função mAP do Darknet retorna o nível de confusão do algoritmo. Das detecções realizadas, 1288 foram casos de verdadeiros positivos (TP), contra 230 casos de falsos positivos (FP). Além disso, o modelo deixou de prever 503 instâncias de animais (FN).

Na figura 6 podemos ver a detecção e classificação para uma das imagens utilizadas. O Darknet atribui uma cor para as caixas delimitadoras de acordo com a classe, além de atribuir no canto superior esquerdo a porcentagem de certeza daquele objeto detectado ser da classe descrita.

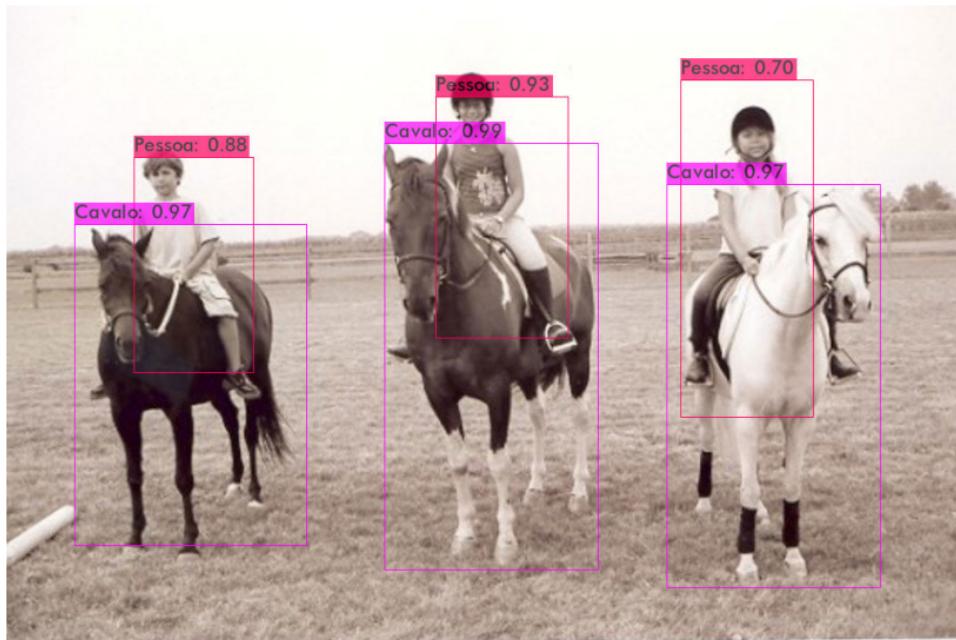


Figura 6. Detecção e classificação de Pessoas e Cavalos em uma imagem.

Importante notar a baixa precisão para a classe Pessoa. Esta classe obteve um baixo desempenho, devido ao grande número de instâncias, como observado na tabela 1. Muitas destas pessoas estavam em diferentes posições e contextos. Por exemplo, em imagens adquiridas da classe Cavalo, muitas pessoas estavam montadas sobre estes animais, enquanto em imagens com cachorros ou gatos, muitos apareciam fora de foco ou agachadas próximas aos animais.

Esta variedade de posições são uma das principais causas do baixo desempenho do modelo, mas como o foco era a detecção de animais, a classe Pessoa não sofreu um tratamento para aumentar a sua precisão.

6. Conclusão

Este trabalho apresentou um modelo de sistema de detecção de objetos em tempo real utilizando o YOLO, aplicado à detecção de animais como forma de monitoramento através de vídeos em câmeras de segurança. Dessa forma, o projeto apresentado mostrou que é possível minimizar os riscos e como principal objetivo desenvolver um sistema que apresenta eficiência na detecção desses animais.

O resultado obtido de 84.58% para a precisão média do modelo é considerado satisfatório tendo em vista a complexidade das diferentes classes propostas, onde algumas possuíam características bem semelhantes.

O modelo apresentado foi capaz de mostrar que o seu uso para a detecção e reconhecimento de animais é eficaz, trazendo resultados bem significativos para o contexto, incentivando

o desenvolvimento de sistemas mais complexos, aumentando a possibilidade de resolver satisfatoriamente a situação problemática com base nas técnicas apropriadas para a melhoria do processo.

7. Trabalhos Futuros

Em trabalhos futuros, este projeto pode ser melhorado com diversas camadas de técnicas de processamento digital de imagens, melhorando a qualidade das imagens dos vídeos, e também o desenvolvimento de trabalhos para imagens obtidas com infravermelho, uma vez que muitos animais silvestres possuem hábitos noturnos.

A detecção de movimento também poderá trazer uma melhoria significativa para melhorar o desempenho, uma vez que a detecção e classificação do objeto só ocorreria após a identificação de movimento no ambiente e na área em que ocorreu o movimento, reduzindo o custo computacional.

Outra implementação que a ser realizada é a previsão do tamanho do animal detectado de acordo com sua distância da câmera, pois facilitará a distinção, por exemplo, entre gatos e outros felinos, onde o nível de alarde na detecção poderá ser melhorado.

Referências

- [Adami et al. 2021] Adami, D., Ojo, M. O., and Giordano, S. (2021). Design, development and evaluation of an intelligent animal repelling system for crop protection based on embedded edge-ai. *IEEE Access*, 9:132125–132139.
- [Bochkovskiy et al. 2020] Bochkovskiy, A., Wang, C., and Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934.
- [Garg et al. 2018] Garg, D., Goel, P., Pandya, S., Ganatra, A., and Kotecha, K. (2018). A deep learning approach for face detection using yolo. In *2018 IEEE Punecon*, pages 1–4.
- [Girshick 2015] Girshick, R. (2015). Fast r-cnn. *CoRR*, abs/1504.08083.
- [Hussain et al. 2019] Hussain, M., Bird, J. J., and Faria, D. R. (2019). A study on cnn transfer learning for image classification. In Lotfi, A., Bouchachia, H., Gegov, A., Langensiepen, C., and McGinnity, M., editors, *Advances in Computational Intelligence Systems*, pages 191–202, Cham. Springer International Publishing.
- [Karn 2021] Karn, A. (2021). Artificial intelligence in computer vision. *International Journal of Engineering Applied Sciences and Technology*, 6:249–254.
- [Kathuria 2018] Kathuria, A. (2018). What's new in yolo v3. <https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>. Accessed: 2021-10-18.
- [Krasin et al. 2017] Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Kamali, S., Mallochi, M., Pont-Tuset, J., Veit, A., Belongie, S., Gomes, V., Gupta, A., Sun, C., Chechik, G., Cai, D., Feng, Z., Narayanan, D., and Murphy, K. (2017). Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*.
- [Kuznetsova et al. 2020] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Mallochi, M., Kolesnikov, A., Duerig, T., and Ferrari, V. (2020). The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*.

- [LeCun et al. 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–44.
- [Moallem et al. 2021] Moallem, G., Pathirage, D. D., Reznick, J., Gallagher, J., and Sari-Sarraf, H. (2021). An explainable deep vision system for animal classification and detection in trail-camera images with automatic post-deployment retraining. *Knowledge-Based Systems*, 216:106815.
- [Redmon 2016] Redmon, J. (2013–2016). Darknet: Open source neural networks in c. <http://pjreddie.com/darknet/>.
- [Redmon et al. 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.
- [Redmon and Farhadi 2017] Redmon, J. and Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525.
- [Redmon and Farhadi 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, abs/1804.02767.
- [Schneider et al. 2018] Schneider, S., Taylor, G. W., and Kremer, S. (2018). Deep learning object detection methods for ecological camera trap data. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 321–328.
- [Tan et al. 2018] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A survey on deep transfer learning. *CoRR*, abs/1808.01974.
- [Turečková et al. 2020] Turečková, A., Holik, T., and Oplatkova, Z. (2020). Dog face detection using yolo network. *MENDEL*, 26:17–22.