http://journal.unoeste.br/index.php/ce/index
DOI: 10.5747/ce.2021.v13.n2.e358
ISSN on-line 2178-8332

Colloquium Exactarum

Submetido: 13/03/2021 Revisado: 30/05/2021 Aceito:01/08/2021

# DETECÇÃO DE ANIMAIS BOVINOS UTILIZANDO IMAGENS AÉREAS POR MEIO DE REDES NEURAIS

## Detection of Cattle using aerial images with Neural Network

Wellington Hiroshi Takano, Leandro Luiz de Almeida, Francisco Assis da Silva

Universidade do Oeste Paulista
Faculdade de Informática de Presidente Prudente - FIPP
wellz-tkn@hotmail.com, llalmeida@unoeste.br, chico@unoeste.br

**RESUMO –** Atualmente com a evolução da tecnologia, muitas áreas que abrangem a agropecuária estão aderindo ao uso de drones profissionais que possuem diversas ferramentas que auxiliam no monitoramento. Com o avanço das Redes Neurais, diversos pesquisadores estão optando em utilizar redes neurais para efetuar detecção de objetos. Como a contagem de animais bovinos requer tempo e esforço físico, além de ser arriscado em determinadas situações, com a utilização de imagens aéreas e de redes neurais, essa atividade torna-se mais viável e com um gasto de tempo menor. Neste trabalho o foco está em detectar animais bovinos utilizando duas redes neurais, com imagens aéreas capturadas por meio de um drone.
**Palavras-chave:** Detecção de objeto, Redes Neurais, Detecção de gado, Drone, CNN.

**ABSTRACT –** Nowadays, with the evolution of technology, many areas that cover agriculture are making use of professional drones that have several tools that help in monitoring. With the advancement of Neural Networks, several researchers are choosing to use neural networks to detect objects. As the counting of bovine animals requires time and physical effort, in addition to being risky in certain situations, with the use of aerial images and neural networks, this activity becomes more viable and with less time spent. In this work, the focus is on detecting bovine animals using two neural networks, with aerial images captured from a drone.
**Keywords:** Object Detection, Neural Network, Cattle detection, Drone, CNN.

## 1. INTRODUÇÃO

Muitas indústrias têm como objetivo obter mais dados a respeito das suas operações de negócios e os fazendeiros não são diferentes. Com o avanço da agricultura de precisão (MCBRATNEY et al, 2005), muitos fazendeiros demandam por mais dados sobre tópicos, tais como: temperatura do solo, umidade e monitoramento individual do rebanho para determinar o crescimento e bem-estar dos bovinos. Porém, obter essas informações instantâneas e precisas sobre a localização de seus animais torna-se inacessível em grandes fazendas (SARWAY, 2018). Andar pelo pasto por diversas horas contando o rebanho é uma tarefa exaustiva. Portanto, o monitoramento do rebanho utilizando a computação visual, além de ser de baixo custo e evitar serviços exaustivos, é um método promissor para essa tarefa ( SHAO, 2019).

As imagens aéreas obtidas por um Veículo Não Tripulado (UAV) mostram algo extremamente promissor para monitorar animais em vastas áreas que são difíceis de serem

monitoradas pelo solo. Uma das restrições mais importantes dessa tarefa é a necessidade de desenvolver e aplicar sistemas automatizados de detecção em imagens. Veículos não tripulados provaram ser efetivos no monitoramento da vida selvagem, mas para os veículos não tripulados serem ferramentas de monitoramento eficientes, é preciso uma coleção de dados para analisar e melhorar a capacidade de automatizar a detecção de animais e contagem (GONZALEZ et al., 2016) (OISHI, MATSUNAGA, 2014).

Com o avanço da *Deep Learning*, e particularmente com a Redes Neurais em aplicações com Visão Computacional, o reconhecimento de objetos e acurácia na classificação alcançou um nível de melhorias impressionante. A evolução das Unidades de Processamentos Gráficos (GPU) também foi um fator que contribuiu significativamente para adoção das Redes Neurais na Visão Computacional, por lidar com o problema de processamento visual em tempo real com tarefas intensivas em paralelo (BENJDIRA et al., 2019).

No ano de 2018 a YOLOv3 foi lançada, sendo caracterizada pela alta acurácia e substituição da função *softmax* com regressão logística e *threshold* (BENJDIRA et al, 2019). A YOLOv3 além de ter uma precisão desejável e alta velocidade, também possui um bom desempenho para detectar objetos pequenos (TIAN et al, 2019).

Este trabalho contribui com uma metodologia de detecção de animais bovinos utilizando imagens aéreas obtidas por drone. Para isso foi utilizado um *dataset* disponibilizado pela Universidade de Tokyo (Dataset, 2020). Durante as etapas dos processos, foi realizado um pré-processamento nas imagens e *data augmentation*, pois a quantidade de imagens no *dataset* disponível era insuficiente para treinar a rede. Para realizar o treinamento, foi utilizado o serviço gratuito disponibilizado pela Google denominado Google Colab (Google Colab, 2020) desenvolvido para aqueles que trabalham com área de IA (Inteligência Artificial). É possível treinar a rede em apenas por 12 horas, após isso é necessário um intervalo de espera para que possa utilizar o Google Colab novamente para treinar.

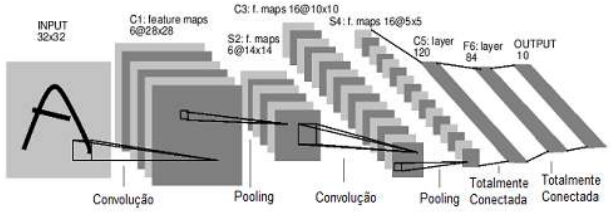
Após esta seção de introdução, o trabalho está organizado da seguinte maneira. Na Seção 2 são tratados os conceitos básicos sobre funcionamento das Redes Neurais Convolucionais. Na Seção 3 é apresentado o funcionamento da rede neural YOLO utilizada para realizar as detecções de bovinos. Na Seção 4, são apresentados os métodos utilizados na etapa inicial até a etapa final detalhada de maneira particionada. Na Seção 5 são apresentados os resultados obtidos por meio da realização dos experimentos. Por fim, na Seção 6 encontram-se as considerações finais do trabalho.

## 2. REDES NEURAIS CONVOLUCIONAIS

A Rede Neural Convolucional (CNN - *Convolutional Neural Network*) se tornou algo essencial na área da Visão Computacional, além de ser fácil de treinar quando se trata de grande quantidade de amostras rotuladas que representam as diferentes classes-alvo. Algumas das vantagens da Rede Neural Convolucional são: extrair características relevantes por meio do aprendizado de transformações (*Kernels*), e depender da quantidade menor possível de ajustes nos parâmetros do que as redes totalmente conectadas, com o mesmo número de camadas ocultas. Devido ao fato de que cada unidade de uma determinada camada não é conectada com todas as camadas posteriores, há como consequência menos pesos a serem atualizados, no entanto, facilitando o treinamento da rede (ARAÚJO et al, 2017).

Um dos pioneiros na utilização de uma Rede Neural Convolucional foi a LeNet (Figura 1), onde a rede foi utilizada para reconhecer dígitos. A Rede Neural Convolucional é uma rede de multicamadas evoluída das multicamadas conhecidas como o *perceptron*. A rede neural convolucional é composta principalmente por várias camadas, que são denominadas como: de entrada, convolucional, *pooling*, totalmente conectada e de saída. Com a Rede Neural Convolucional é possível extrair características e mapeamento com um treinamento da rede mais rápida, além de ter alta acurácia, é frequentemente utilizada para classificações e previsões (SHEN; WANG, 2018) (LECUN et al, 1989).

**Figura 1.** Ilustração da arquitetura LeNet com a imagem de entrada e as camadas: convolucionais, *pooling* e totalmente conectadas.

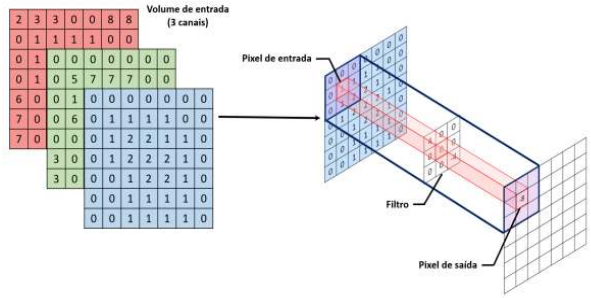



Fonte: (LECUN et al, 1989).

## 2.1 Camada de Convolução

As camadas convolucionais possuem um conjunto de filtros. Nesta camada é feito o mapeamento de características mais relevantes da imagem. Cada filtro possui dimensão reduzida, porém, ele se estende por toda a profundidade do volume de entrada. Por exemplo, se a imagem for colorida, então ela possui três canais (R, G, B) e o filtro da primeira camada convolucional terá o tamanho 5 x 5 x 3 (cinco de largura, cinco de altura e três de profundidade) conforme a Figura 2 (ARAÚJO et al, 2017). Automaticamente, durante o processo de treinamento da rede, esses filtros são ajustados para que sejam ativados em presença de características relevantes identificadas no volume de entrada, como orientação de bordas ou manchas de cores (KARPATHY, 2018).

**Figura 2.** Ilustração da convolução com um filtro 3x3 e o volume de entrada.

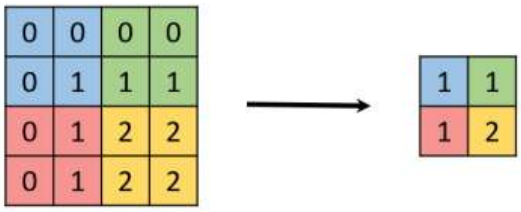



Fonte: (ARAÚJO et al, 2017).

## 2.2 Camada de Pooling

Após uma camada convolucional, geralmente existe uma camada de *pooling*. A função dessa camada é reduzir progressivamente a dimensão espacial do volume de entrada, consequentemente a redução diminui o custo computacional da rede para evitar o overfitting (KARPATHY, 2018). A forma mais comum de *pooling* consiste em substituir os valores de uma região pelo valor máximo (GOODFELLOW et al., 2016). Essa operação é conhecida como *max pooling* e é útil para eliminar valores desprezíveis, reduzindo a dimensão da representação dos dados e acelerando a computação necessária para as próximas camadas, além de criar uma invariância e pequenas mudanças e distorções locais (Figura 3) (ARAÚJO et al, 2017). Outras funções de *pooling* comumente usadas são a média, a norma L2 e a média ponderada baseada na distância, partindo do pixel central (AGGARWAL et al., 2001).

**Figura 3.** Aplicação do *max pooling* em uma imagem 4x4 utilizando um filtro de 2x2.

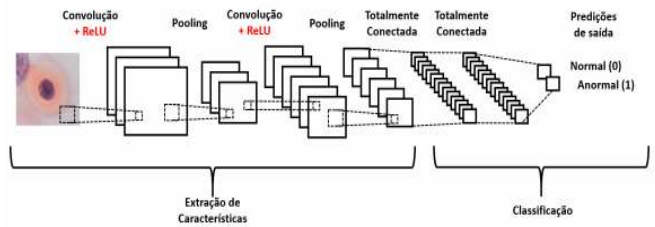



Fonte: (Araújo et al, 2017).

## 2.2 Camada Totalmente Conectada

A saída das camadas convolucionais e de *pooling* representam as características extraídas da imagem de entrada. O objetivo das camadas totalmente conectadas é utilizar suas características para classificar a imagem em uma classe pré-determinada, como ilustrado na Figura 4 (ARAÚJO et al, 2017).

**Figura 4.** Ilustração da extração de características de uma imagem por uma Rede Neural Convolucional e sua posterior classificação.




Fonte: (Araújo et al, 2017).

Essas camadas são formadas por unidades de processamento conhecidas como neurônios, e o termo “totalmente conectado” significa que todos os neurônios da camada anterior estão conectados a todos os neurônios da camada posterior (ARAÚJO et al, 2017).

## 3. YOLO

A YOLOv3 é a rede mais comum entre todas as redes com rapidez e acurácia (LIU et al., 2019). A YOLOv3 contém ao todo 53 camadas

convolucionais como apresentado na Figura 5. Com esta arquitetura, houve melhorias tanto na acurácia da detecção de objetos quanto na otimização do uso da GPU, tornando-se mais eficiente o uso computacional (FACHRIE, 2020).

A diferença entre a YOLOv4 e a YOLOv3 é o *backbone*. Enquanto a YOLOv3 utiliza *Darknet53* a YOLOv4 utiliza a *CSPDarknet53* (REDMON; FARHADI, 2018) (BOCHKOVSKIY; WANG; LIAO, 2020).

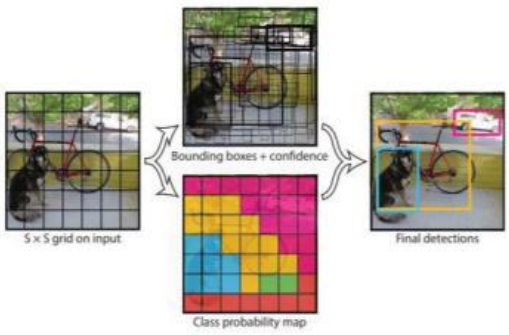
**Figura 5.** Arquitetura da YOLOv3 (Redmon et al, 2018).

| | Type | Filters | Size | Output |
|---|---|---|---|---|
| | Convolutional | 32 | 3 × 3 | 256 × 256 |
| | Convolutional | 64 | 3 × 3 / 2 | 128 × 128 |
| 1× | Convolutional | 32 | 1 × 1 | |
| | Convolutional | 64 | 3 × 3 | |
| | Residual | | | 128 × 128 |
| | Convolutional | 128 | 3 × 3 / 2 | 64 × 64 |
| 2× | Convolutional | 64 | 1 × 1 | |
| | Convolutional | 128 | 3 × 3 | |
| | Residual | | | 64 × 64 |
| | Convolutional | 256 | 3 × 3 / 2 | 32 × 32 |
| 8× | Convolutional | 128 | 1 × 1 | |
| | Convolutional | 256 | 3 × 3 | |
| | Residual | | | 32 × 32 |
| | Convolutional | 512 | 3 × 3 / 2 | 16 × 16 |
| 8× | Convolutional | 256 | 1 × 1 | |
| | Convolutional | 512 | 3 × 3 | |
| | Residual | | | 16 × 16 |
| | Convolutional | 1024 | 3 × 3 / 2 | 8 × 8 |
| 4× | Convolutional | 512 | 1 × 1 | |
| | Convolutional | 1024 | 3 × 3 | |
| | Residual | | | 8 × 8 |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

Fonte: (REDMON et al, 2018).

A YOLOv3 divide a imagem em regiões e projeta as caixas delimitadoras e a probabilidade de cada região (Figura 6). A YOLOv3 prevê uma pontuação do objeto utilizando regressão logística (HASSAN et al, 2020).

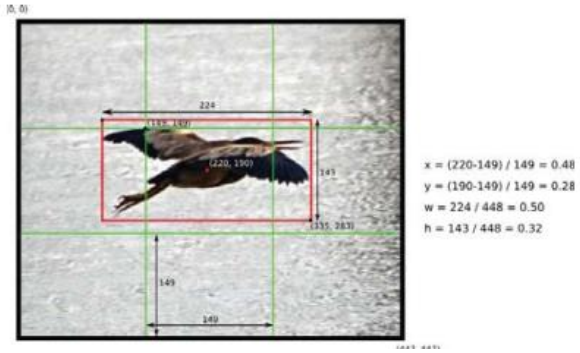
**Figura 6.** Funcionamento da YOLOv3.




Fonte: (HASSAN et al, 2020).

Inicialmente, a rede divide as imagens em regiões, divide-as em *grids* SxS e, após esse processo, projeta as caixas delimitadoras na imagem. As caixas delimitadoras são ponderadas pela probabilidade e o modelo faz as detecções com base nos pesos finais (Figura 7) (HASSAN et al., 2020).

**Figura 7.** Calculando os valores do *bounding box.*




Fonte: (HASSAN et al, 2020).

Há 5 valores na caixa delimitadora: *x*, *y*, *w*, *he* e confiança. A posição das caixas delimitadoras é representada pelas coordenadas (*x*, *y*), que são normalizadas para [0,1]. A Figura 7 mostra também os valores de *w* e *h* na qual representam as dimensões da caixa em relação ao tamanho da imagem, que também são normalizados para [0,1]. Por fim, a confiança representa a presença ou ausência do objeto desejado. Caso não haja nenhum objeto dentro daquela célula, a pontuação será de 0. Se houver um objeto, a pontuação da confiança será equivalente à intersecção sobre união (IoU) entre a caixa prevista e a verdadeira.

No entanto, cada célula da grade projeta B caixa delimitadoras, na qual resulta em S x S x B * 5 nas saídas dentro da caixa delimitadora. Ao adicionar as saídas do vetor obtém-se S x S x (B * 5 + C) como o resultado (MENEGAZ, 2018).

## 4 AQUISIÇÃO DO *DATASET*

Para o desenvolvimento deste trabalho, foram utilizados 2 *datasets* para realizar o treinamento e testes. Desses dois *datasets*, um foi utilizado para realizar o treinamento e o outro para testar a robustez da rede neural.

O *dataset* é dividido em duas partes, no qual as imagens foram obtidas do mesmo pasto, porém, em datas diferentes. O primeiro *dataset* foi obtido em 24 de Maio de 2016 e o segundo em 24 de agosto de 2016. Ambos os *datasets* possuem imagens capturadas com um DJI Phantom 4, pesando 1380g, com um voo máximo

de 28 minutos. Este drone possui a capacidade de gravar vídeos, mas foram utilizadas apenas imagens que possuem resolução 4000x3000 *pixels*.

O *dataset* 1 contém 656 imagens e o clima no local na qual foram obtidas as imagens estava ensolarado e com o céu limpo. Durante o voo o drone foi mantido numa altura de 50 metros que cobria uma área de 60 x 80 metros (SHAO, 2019). O *dataset* 2 foi obtido em outro dia, mas com o mesmo tipo de bovino, contendo ao todo 14 imagens, neste dia o clima estava ensolarado e limpo. Esse *dataset* foi obtido no mesmo pasto em Kumamoto, Japão.

O *dataset* foi separado em duas partes: imagens para realizar o treinamento e o restante para testes. Para realizar o treinamento foram selecionadas 302 imagens e para testes 46. A média do tamanho dos animais bovinos é de 87 x 90 pixels presentes no *dataset* 1 e 59 x 101 pixels no *dataset* 2 ( SHAO, 2019).

De todas as imagens selecionadas para realizar o treinamento, foram contabilizados ao todo 1271 animais bovinos presentes na imagem, além disso, em alguns casos ocorria de o mesmo bovino estar presente em outras imagens diferentes.

### 4.1 Pré-Processamento

Para realizar o treinamento, foi necessário realizar um pré-processamento das imagens. Devido ao fato da resolução do *dataset* ser muito alta e a rede neural não permitir esta resolução, houve a necessidade de recortar das imagens em tamanhos menores para tornar possível o treinamento da rede e acelerar o treinamento. Portanto, as imagens foram recortadas em duas dimensões diferentes na qual são de 512 x 512 *pixels* e 768 x 768 *pixels*.

A partir das imagens do *dataset* recortadas em 512 x 512 *pixels*, foram geradas 692 imagens e com 768 x 768 *pixels*, 542 ao todo.

Após o término desse processo de recorte das imagens em tamanhos menores, foi necessário realizar a seleção das regiões de interesse. Para realizar esta etapa, foi utilizado um software *open source* denominado LabelImg (LabelImg, 2020). O LabelImg é uma ferramenta de notações gráficas onde pode-se demarcar as regiões de interesses tanto para YOLO quanto para outras redes, conforme mostrado na Figura 8. Além disso, foram retiradas todas as imagens que os bovinos continham sombras, deixando apenas os demais no qual não tinham sombras como na Figura 9.

**Figura 8.** Regiões de interesse.



Fonte: (Autor, 2020).

### 4.2 *Data Augmentation*

Como a quantidade de *dataset* estava escassa, foi necessário realizar o processo de *data augmentation* (aumento de dados), na qual é aplicado um processamento de imagem no *dataset* para conseguir aumentar a quantidade de imagens. Essas aplicações de processamento de imagem podem rotacionar as imagens em diversos ângulos, aplicar borramento, saturação, espelhamento, brilho entre outros.

No entanto, foi utilizado o Roboflow (Roboflow, 2020), que é uma ferramenta de organização e gerenciamento do *dataset*. O Roboflow possui diversas ferramentas para auxiliar no gerenciamento entre elas o *data augmentation*.

Nos *datasets* utilizados no trabalho, foram aplicadas rotações em 90 graus no sentido horário, anti-horário e espelhamento, como pode ser visto na Figura 9.

**Figura 9.** As rotações aplicadas.




Fonte: (Autor, 2020).

A Tabela 1 mostra a quantidade de imagens dos dois *datasets*, antes e após o aumento de dados.

**Tabela 1.** Quantidade de imagens antes e depois do *data augmentation*.

| **Resolução** | **Pré-*Augmentation*** | **Pós-*Augmentation*** |
|---|---|---|
| 512 x 512 | 692 | 2076 |
| 768 x 768 | 542 | 1626 |

Fonte: (Autor, 2020).

### 4.3 Treinamento

Para realizar o treinamento da rede foi utilizado o Google Colab (Google Colab, 2020), na qual é uma ferramenta disponibilizada gratuitamente pela Google para trabalhar com a área de IA e *Machine Learning*.

Para iniciar o treinamento da rede neural, foi preciso modificar as configurações do arquivo .cfg, onde é configurada a quantidade de iterações, filtros e etc.

A principal variável a serem configurada foi a quantidade de filtros necessários, que para isso existe uma fórmula para definir a quantidade de filtros de acordo com a quantidade de classes que se deseja detectar.

$$filters = (classes + 5) * 3 \quad (1)$$

Portanto, a quantidade de filtros será 18, pois, utilizou-se apenas com uma única classe. A largura e a altura, na configuração foram de acordo com a resolução das imagens do *dataset*. Quando a rede foi treinada com a resolução em 512 x 512 *pixels*, a largura e a altura foram configuradas em 512. O valor de *batch* foi de 64 e subdivisão em 32 para ambas as redes treinadas.

Na rede neural YOLOv3, a quantidade de iterações é diferente da YOLOv4, portanto, na YOLOv3 a quantidade de iterações é definida por uma simples fórmula.

$$Iterations = 2000 * classes \quad (2)$$

Como foi trabalhado apenas com uma classe, pela formula a quantidade de iterações deveria ser de 2000, porém, a própria YOLOv3 não recomenda treinar com a quantidade de *datasets* maior que a quantidade iterações. Com a YOLOv4 a quantidade de iterações foi um pouco maior do que a treinada com a YOLOv3, na YOLOv4 foi aplicada 6000 iterações ao invés de 4000 como na YOLOv3.

### 4.4 Google Colab

O Google oferece uma ferramenta gratuita, o Google Colab (Google Colab, 2020), para aqueles que queiram trabalhar com a área de *machine learning*. Portanto, ela fornece recursos poderosos como uma GPU virtual que pode ser utilizada por 12 horas seguidas. Por ser uma ferramenta gratuita, o Google distribui esses recursos de forma igualitária para que todos possam usufruir desses recursos.

As GPUs utilizadas para treinar a rede não possibilitam seleção, portanto o fornecimento da GPU possui algumas variáveis, como o quão o usuário usou e o tempo utilizado. No entanto, nem todas as redes foram treinadas com uma única GPU.

As GPUs disponibilizadas pelo Google Colab são Nvidia K80s, T4s, P4s e P100s. Na maior parte das vezes, as redes foram treinadas utilizando a Tesla T4, que é uma GPU poderosíssima, com 16 GB de RAM.

### 4.5 Detecção

Para realizar as detecções, foi necessário reconfigurar o arquivo .cfg. O valor de *batch* e *subdivision* foram configurados para 1 para realizar as detecções. Além disso, como os animais bovinos nas imagens ficavam pequenos, e na própria documentação da YOLOv3 diz que

para objetos muito pequenos os valores da altura e largura devem ser maiores para detectar objetos pequenos, esses valores também foram alterados.

Foram realizados alguns experimentos de detecção, tais como aumentar gradativamente o valor da altura e largura do arquivo .cfg para se analisar quais são os melhores valores obtidos.

## 5. RESULTADOS OBTIDOS

Foram realizados testes com duas redes neurais diferentes, sendo elas: YOLOv3 e YOLOv4. Com a YOLOv3 foi realizado o treinamento com duas resoluções diferentes, já com a YOLOv4 foi possível realizar o treino com apenas uma dimensão devido a alguns problemas durante o treino da rede.

Foram feitas as comparações da YOLOv3 com a YOLOv4, ambas treinadas da mesma forma e com o mesmo arquivo de configuração no momento de efetuar as detecções.

Na Tabela 2 tem-se os resultados obtidos das 2 dimensões diferentes treinadas e a acurácia de cada uma delas.

**Tabela 2.** Taxa de acertos com a YOLOv3 com as imagens utilizadas para detectar em 2560x2560.

| | **Dataset 1** | **Dataset 2** |
|---|---|---|
| 512x512 | 98,22% | 69,64% |
| 768x768 | 96,80% | 83,92% |

Fonte: (Autor, 2020).

Na Tabela 3 tem-se a taxa de acurácia onde foi utilizado a rede treinada com o *dataset* em 512 x 512 e a imagem para realizar as detecções com a resolução em 2048 x 2048.

**Tabela 3.** Comparação com a taxa de acurácia entre a YOLOv3 e YOLOv4 com a rede treinada em 512 x 512 e a imagem utilizada para realizar detecção com a resolução em 2048 x 2048

| | **Dataset 1** | **Dataset 2** |
|---|---|---|
| YOLOv3 | 95,39% | 62,50% |
| YOLOv4 | 95,55% | 92,85% |

Fonte: (Autor, 2020).

Durante a realização da detecção, alguns problemas foram encontrados, principalmente no *dataset* 2 onde há vários pontos de sombra e objetos escuros, como pode ser observado na Figura 10.

**Figura 10.** Alguns problemas encontrados durante a detecção.




Fonte: (Autor, 2020).

Na figura 10, é possível notar alguns falso-positivos (FP) onde a YOLO detecta uma sombra e um carro como um animal bovino. Ao analisar o *dataset* 1*,* nota-se que há muitos animais bovinos pretos e que a YOLO pode estar confundindo a sombra com um animal bovino preto.

Outro detalhe importante a ressaltar é que no mesmo *dataset* é observada a presença de um animal bovino com uma sombra, obtendo a mesma característica que a do veículo que foi detectado na Figura 10.

**Figura 11.** Imagem do *dataset 1* com um animal bovino com sombra.



Fonte: (Autor, 2020).

Como visto na Figura 11, todos os bovinos no qual apresentavam sombras no *dataset* como o da figura foram removidos.

Apesar de ter removido todos os animais bovinos com sombra, ainda assim, alguns veículos eram detectados, porém com valores menores de taxa de precisão.

**Figura 12.** Imagem do *dataset* 1 com um bovino preto deitado.



Fonte: (Autor, 2020).

No entanto, no *dataset* 1, além dos animais com sombra, é possível notar uma semelhança dos bovinos pretos deitados com os carros detectados conforme a Figura 12.

### 5.1. *Precision, Recall* e *F-measure*

Para calcular a precisão, *recall* e *f-measure* utiliza-se as seguintes fórmulas:

$$Precision = TP / TP + FP \quad (3)$$

$$Recall = TP / TP + FN \quad (4)$$

$$F\text{-}Measure = 2 * Precision * Recall / Precision + Recall \quad (5)$$

O verdadeiro-positivo (TP) é quando o objeto detecta corretamente, o falso-positivo quando detecta algo que não são animais bovinos. Os falso-negativos (FN) é a quantidade de animais bovinos que não foram detectados.

Na Tabela 4, a rede treinada com o dataset em 512 x 512 apresentaram bons resultados com o dataset 1 de 0.97 de precisão, 0.98 o *recall com f-measure de 0,97* enquanto com o dataset 2 foram obtidos 0.38 de precisão, 0.58 de *recall* com 0.97 de *f-measure*. Com o dataset treinado em 768 x 768, foram obtidos uma precisão de 0.96, 0.98 de recall e 0.96 de f-measure com o dataset 1 e 0.60 com dataset 2. Em comparação com 512 x 512 e 768 x 768, não houve muita diferença de resultados no dataset 1, diferente do dataset 2 no qual foram obtidos uma relevância de 0.22 em relação do dataset 2.

**Tabela 4.** Valores de Precisão com a YOLOv3.

| | **Dataset 1** | **Dataset 2** |
|---|---|---|
| 512x512 | 0.97 | 0.38 |
| 768x768 | 0.96 | 0.60 |

Fonte: (Autor, 2020).

**Tabela 5.** Valores do *Recall* com a YOLOv3.

| | **Dataset 1** | **Dataset 2** |
|---|---|---|
| 512x512 | 0.98 | 0.58 |
| 768x768 | 0.98 | 0.63 |

Fonte: (Autor, 2020).

**Tabela 6.** F-Measure com a YOLOv3.

| | **Dataset 1** | **Dataset 2** |
|---|---|---|
| 512x512 | 0.97 | 0.97 |
| 768x768 | 0.96 | 0.96 |

Fonte: (Autor, 2020).

Os verdadeiros positivos são os animais bovinos detectados corretamente enquanto os falso-positivos são quando detectam o objeto que não é daquela classe.

Na Tabela 7 é apresentada a comparação da precisão de ambas as redes onde foram utilizadas imagens com resolução em 4000 x 3000. Ao rodarmos a Rede Neural Convolucional para realizar as detecções, a própria YOLO redimensiona a imagem para 2048 x 2048 no qual foi a maior resolução possível com a YOLOv4 e com a YOLOv3 2560 x 2560.

**Tabela 7.** Taxa de precisão com a YOLOv3 e YOLOv4.

| | **Dataset 1** | **Dataset 2** |
|---|---|---|
| 512x512 | 0.94 | 0.41 |
| 768x768 | 0.92 | 0.58 |

Fonte: (Autor, 2020).

## 5. CONSIDERAÇÕES FINAIS

Neste trabalho foram apresentadas as duas Redes Neurais Convolucionais e como foram realizados os processos até a detecção dos bovinos. Analisando os resultados de ambas as redes, foram obtidos bons resultados como

exibido nas tabelas. É notado que a YOLOv3 é uma ótima rede neural para detectar objetos muito pequenos. Nas versões anteriores, a YOLO tinha problemas em detectar objetos pequenos. Entretanto, nas versões mais recentes é ao contrário, pois para objetos pequenos em termos de desempenho é bom, mas em objetos médios e grandes é inferior (REDMON; FARHADI, 2018).

Fazendo uma comparação nas duas redes, ambas treinadas da mesma forma e utilizando as mesmas configurações de detecção, verifica-se bons resultados tanto na YOLOv3 quanto na YOLOv4 tendo uma taxa de acurácia de 95,39% com a YOLOv3 e 95,55% no *dataset* 1 onde não houve uma diferença significativa nos resultados, diferente do dataset 2 a YOLOv3 obteve uma taxa de 62,50% de acurácia e a YOLOv4 de 92,85%.

Ao analisar a detecção realizada com as imagens nas resoluções 2048 x 2048 e 2560 x 2560 com a YOLOv3, nota-se uma diferença maior da acurácia com a resolução em 2560 x 2560 em relação a 2048 x 2048 quanto utilizada a rede treinada com as imagens em 512x512. Portanto, quando aumentada a resolução da imagem a ser realizada a detecção, percebe-se um aumento na acurácia.

Com a YOLOv4 não foi possível realizar o treinamento com as imagens com a resolução em 768x768, pois durante o treinamento estourava a memória RAM(Memória de Acesso Aleatório). Além disso, ao realizar as detecções nas imagens, a maior resolução possível da imagem foi de 2048x2048. O Google Colab aloca 12GB de memória *RAM* para utilizarmos. Ao aplicar uma resolução maior, ocorria de estourar a memória, pois quanto maior a resolução, maior era o consumo de memória. Portanto, devido a falta de recursos não foi possível testar outras resoluções maiores.

Com as detecções realizadas no *dataset* 2 onde foi tiradas no mesmo local, porém em estações diferentes, a YOLO classificava objectos como sombras e veículos como um bovino escuro.

Portanto, conclui-se que ambas as redes utilizadas neste projeto tiveram bons desempenhos principalmente para detectar objetos pequenos onde a YOLOv2 tinha alguns problemas em relação a isso. Além disso, ao compararmos ambas as redes, com as mesmas resoluções, a YOLOv4 teve um desempenho melhor em relação a YOLOv3.

## REFERÊNCIAS


AGGARWAL, C., **Outlier detection for high dimensional data**. 2020. Disponível em : https://dl.acm.org/doi/abs/10.1145/375663.375668. Acesso em: 25 out. 2020.

ARAÚJO, F. H. D. *et al.* Redes Neurais Convolucionais com Tensor Flow: Teoria e Prática**.** III ESCOLA REGIONAL DE INFORMÁTICA DO PIAUÍ. 2017. Escola Regional de Informática do Piauí. Disponível em: https://docplayer.com.br/57119969-Redes-neurais-convolucionais-com-tensorflow-teoria-e-pratica.html. Acesso: 25 nov. 2020.

B. BENJDIRA, T. *et al.* Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3. *2*019. *In*: INTERNATIONAL CONFERENCE ON UNMANNED VEHICLE SYSTEMS-OMAN (UVS). 1., Disponíve em: https://ieeexplore.ieee.org/abstract/document/8658300. Acesso em: 25 nov. 2020. https://doi.org/10.1109/UVS.2019.8658300

BOCHKOVSKIY, A.; WANG, C.-Y.;LIAO, H.-Y. M. **YOLOv4:** Optimal Speed and Accuracy of Object. 2020. Detection. Disponível em: https://arxiv.org/abs/2004.10934. Acesso em: 22 nov. 2020.

DATASET, Dataset. 2019. Disponível em: http://bird.nae-lab.org/cattle/ Acesso: 22 nov. 2020.

DATASET, Dataset. 2019. Disponível em : http://bird.nae-lab.org/cattle/. Acesso: 26 nov. 2020.

FACHRIE, M. **A simple vehicle counting system using deep learning with YOLOv3 Model.** 2020. Disponível em : https://www.researchgate.net/publication/341075968_A_Simple_Vehicle_Counting_System_Using_Deep_Learning_with_YOLOv3_Model. Acesso em: 26 out. 2020. https://doi.org/10.29207/resti.v4i3.1871

GONZALEZ, L. F. *et al.* 2016. **Unmanned Aerial Vehicles (Uavs) and Artificial Intelligence Revolutionizing Wildlife Monitoring and Conservation**. 2016. Disponível em: https://www.mdpi.com/1424-8220/16/1/97. Acesso: 25 nov. 2020.

GOODFELLOW, J. **Deep Learning.** 2016.. Disponível em : https://books.google.com.br/books?hl=pt-PT&lr=&id=omivDQAAQBAJ&oi=fnd&pg=PR5&dq=deep+learning+goodfellow+pdf&ots=MNO3gnpzQR&sig=hcU98sXLnC5jYgQBpfMWnDZtPCI#v=onepage&q=deep%20learning%20goodfellow%20pdf&f=false. Acesso em: 22 nov. 2020.

GOOGLE Colab, **Google Colab**. Disponível em : https://colab.research.google.com/notebooks/intro.ipynb. Acesso: 25 nov. 2020.

Hassan,N. *et al.* , **People Detection System Using YOLOv3 Algorithm**. 2019. Disponível em **:** https://ieeexplore.ieee.org/abstract/document/9204925/. Acesso em: 25 nov. 2020. https://doi.org/10.1109/ICCSCE50387.2020.9204925

KARPATHY, A. **CS231n Convolutional Neural Networks for Visual Recognition.** Disponível em: http://cs231n.github.io/convolutional-networks/. Acesso em: 15 out. 2018.

LabelImg. **LabelImg**. Disponível em : https://githm/tzutalin/labelImg. Acesso: 22 maio 2020.

LIU, R. *et al.* **An Improved YOLOV3 for Pedestrian Clothing Detection.** INTERNATIONAL CONFERENCE ON SYSTEMS AND INFORMATICS. 6., 2019. Disponível em: https://ieeexplore.ieee.org/abstract/document/9010512. Acesso em: 15 out. 2020. https://doi.org/10.1109/ICSAI48974.2019.9010512

MCBRATNEY, A. *et al.* **Fu-ture directions of precision agriculture. Precision agricul-ture.** 2006. Disponível em : https://link.springer.com/article/10.1007/s11119-005-0681-8. Acesso em: 18 out. 2020.

MENEGAZ, M., **Understanding YOLO | Hacker Noon**. 2020. Disponível em : https://hackernoon.com/understanding-yolof5a74bbc7967. Acesso em: 18 out. 2020.

REDMON, J., FARHADI, A., **YOLOv3: an Incremental Improvement**. 2018. Disponível em : https://arxiv.org/abs/1804.02767. Acesso em: 23 out. 2020.

ROBOFLOW. **Roboflow**. Disponível em : https://roboflow.com/. Acesso em : 18 out. 2020

SARWAR, F. et al. **Detecting and Counting Sheep with a Convolutional Neural Network**. 2018. Disponível em:https://www.researchgate.net/publication/331106200_Detecting_and_Counting_Sheep_with_a_Convolutional_Neural_Network. Acesso em : 23 nov. 2020. https://doi.org/10.1109/AVSS.2018.8639306

SHAO, W. *et al.* **Cattle Detection and counting in UAV images based on convolutional neural networks**. 2018. Disponível em : https://www.tandfonline.com/doi/full/10.1080/01431161.2019.1624858. Acesso: 22 out. 2020.

SHEN, W.; WANG, W. Node Identification in Wireless Network Based on Convolutional Neural Network. *In*: INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE AND SECURITY. 2018. Disponível em: https://ieeexplore.ieee.org/abstract/document/8564297. Acesso em: 18 out. 2020. https://doi.org/10.1109/CIS2018.2018.00059

Tian, Y. et al. **Detection of Apple Lesions in Orchards Based on Deep Learning Methods of CycleGAN and YOLOV3-Dense**. 2019. Disponível em : https://www.hindawi.com/journals/js/2019/7630926/. Acesso: 11 nov. 2020. https://doi.org/10.1155/2019/7630926