Reconhecimento de Operações de PARE e SIGA pelo Veículo autônomo IARA

Eduardo Frigini, Deivison Vitória

*Abstract*—Nós apresentamos neste trabalho um algoritmo para que o Automóvel Robótico Autônomo Inteligente (IARA) consiga identificar operações de PARE e SIGA durante um trajeto numa via. O IARA é um carro totalmente autônomo que utiliza detecção de objetos para identificar obstáculos. Utilizamos a YOLO que é um sistema de detecção de objetos em tempo real de última geração. Nós utilizamos uma base de imagens de sinalização de PARE e SIGA para treinar a rede e conseguimos bons resultados na detecção destes objetos.

*Index Terms*—PARE e SIGA, YOLO, detecção.

# INTRODUÇÃO

Desde a sua introdução pela LeCun no início dos anos 90, as Convolutions Networks (convets) demonstraram excelente desempenho em tarefas como classificação de dígitos escritos à mão e detecção de rosto. Mais notavelmente, (Krizhevsky et al., 2012) mostra desempenho de batimento recorde no benchmark de classificação ImageNet 2012, com seu modelo convnet atingindo uma taxa de erro de 16,4% (Supervision Network / AlexNet), comparado ao resultado de 2º lugar de 26,1% ( Rede ISI). Atualmente, a rede YOLO tem apresentado resultados incríveis de detecção de imagens e é considerada o estado da arte.   
   Vários fatores são responsáveis ​​por esse interesse renovado nos modelos convnet. Primeiro, a disponibilidade de conjuntos de treinamento muito maiores, com milhões de exemplos rotulados. Em segundo lugar, poderosas implementações de GPU e, por último, melhores estratégias de regularização de modelos, como Dropout (Hinton et al., 2012).   
  Desde então, as empresas vêm utilizando o aprendizado profundo no centro de seus serviços. O Facebook, por exemplo, usa redes neurais para seus algoritmos de marcação automática, o Google para pesquisa de fotos, a Amazon para recomendações de produtos, o Pinterest para personalização de feeds, o Instagram para sua infraestrutura de busca, entre muitas outras aplicações.

II. REVISÃO DA LITERATURA

## Convoluções

As convoluções são uma maneira muito eficiente de extraí-las e aproveitam a estrutura da informação codificada em uma imagem. Supõe-se que os pixels que estão espacialmente próximos juntos irão "cooperar" na formação de uma característica particular de interesse muito mais do que em pixels em cantos opostos da imagem. Além disso, se um determinado recurso (menor) for considerado de grande importância ao definir o rótulo de uma imagem, será igualmente importante se esse recurso for encontrado em qualquer lugar dentro da imagem, independentemente do local.  
    Dada uma imagem bidimensional, eu e uma matriz pequena, K de tamanho h × w (conhecido como um núcleo de convolução), que assumimos codifica uma maneira interessante de um recurso de imagem de extração. Então nós calculamos a imagem convolvida, I ∗ K, sobrepondo o kernel no topo da imagem de todas as formas possíveis, e registrando a soma dos produtos elementares entre a imagem e o kernel:



As imagens abaixo mostram uma visão geral diagramática da fórmula acima e o resultado da aplicação da convolução (com dois kernels separados) sobre uma imagem, para atuar como um detector de borda:

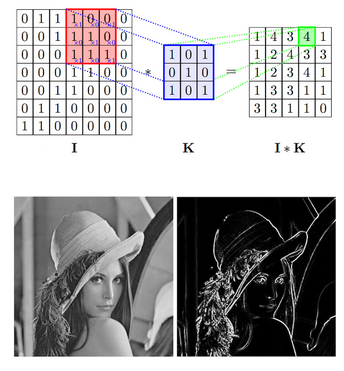


Figura I. Convoluções

## Uma CNN Comum

## Uma arquitetura CNN (Rede Neural Convolucional) típica para uma classificação de imagem de classe k pode ser dividida em duas partes distintas. Primeiro uma cadeia de repetição de camadas convolucionais (às vezes com mais de uma camada convolucional de uma só vez) e segunda seguida por algumas camadas totalmente conectadas (tomando cada pixel das imagens computadas como uma entrada independente), culminando em um k-way camada softmax, para a qual uma perda de entropia cruzada é otimizada. É importante lembrar que após cada camada convolucional ou totalmente conectada, uma ativação (por exemplo, ReLU) será aplicada a todas as saídas.

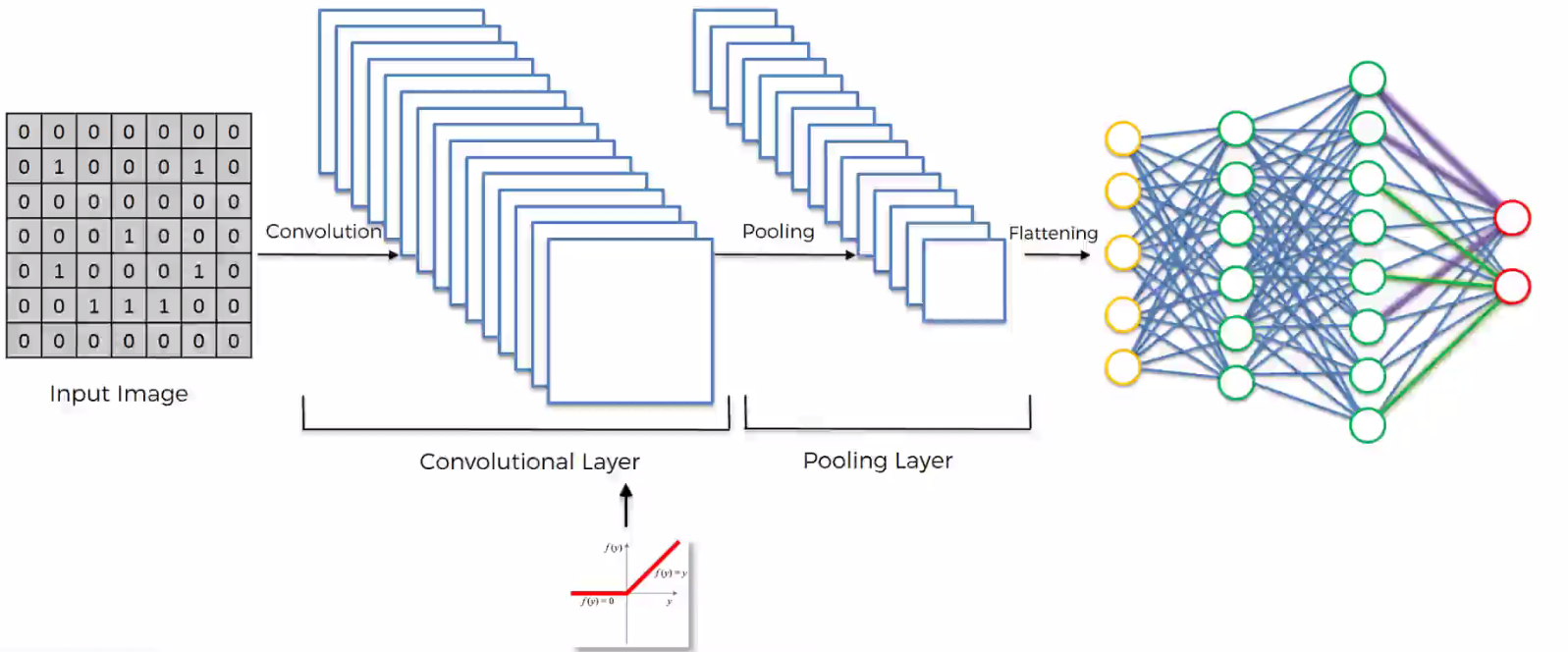


Figura II. Exemplo de rede neural convolucional completa

## Observe o efeito de uma única convolução e passagem de pool através da imagem, reduz a altura e a largura dos canais individuais em favor de seu número, ou seja, profundidade.     O propósito de uma camada softmax é converter qualquer vetor de números reais em um vetor de probabilidades (valores reais não-negativos que somam 1). Nesse contexto, as probabilidades correspondem às probabilidades de que uma imagem de entrada seja um membro de uma determinada classe. Minimizar a perda de entropia cruzada tem o efeito de maximizar a confiança do modelo na classe correta, sem se preocupar com as probabilidades de outras classes. Isto faz com que seja uma escolha mais adequada para tarefas probabilísticas em comparação com, por exemplo, a perda de erro quadrada.

## Processamento de Imagens

A classificação de imagens é a tarefa de obter uma imagem de entrada e produzir uma classe (gato, cachorro, placa de PARE, Placa de SIGA, etc.) ou uma probabilidade de classes que melhor descrevam a imagem. Para os humanos, essa tarefa de reconhecimento é uma das primeiras habilidades que aprendemos desde o momento em que nascemos e ocorre naturalmente e sem esforço quando adultos. Sem sequer pensar duas vezes, conseguimos identificar de forma rápida e transparente o ambiente em que estamos e os objetos que nos rodeiam. Quando vemos uma imagem ou apenas quando olhamos para o mundo ao nosso redor, na maioria das vezes somos capazes de caracterizar imediatamente a cena e dar a cada objeto um rótulo, tudo feito automaticamente pelo cérebro. Essa habilidade de poder reconhecer rapidamente padrões, generalização a partir do conhecimento prévio e adaptar-se a diferentes ambientes de imagem é nossa e não compartilhamos com as máquinas.

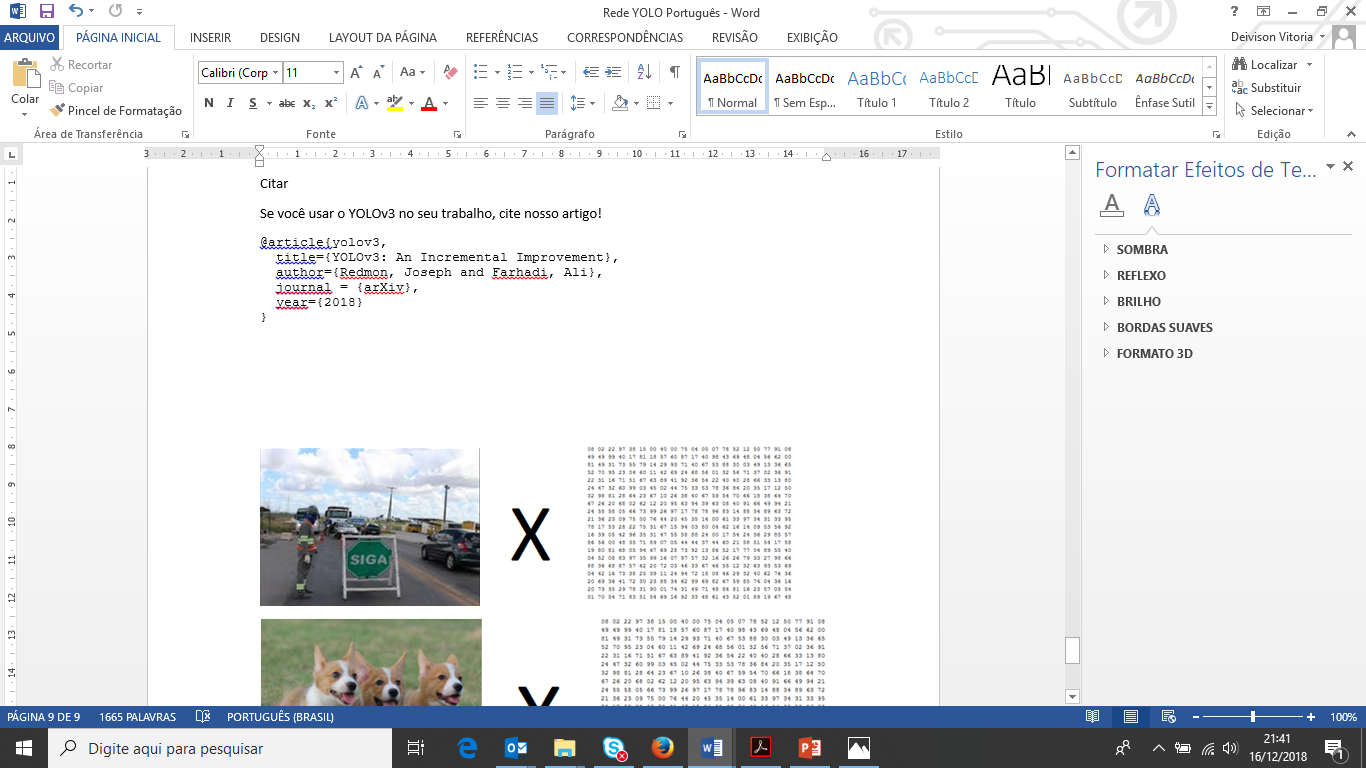


Figura III. O que os humanos enxergam vs. O que os computadores enxergam

Quando um computador vê uma imagem (recebe uma imagem como entrada), ela verá uma matriz de valores de pixel. Dependendo da resolução e do tamanho da imagem, ela verá uma matriz de números de 32 x 32 x 3 (os três se referem a valores RGB). Por exemplo, digamos que temos uma imagem colorida em formato JPG e seu tamanho é 480 x 480. A matriz representativa será 480 x 480 x 3. Cada um desses números recebe um valor de 0 a 255, que descreve o pixel intensidade nesse ponto. Esses números, embora sem sentido para nós quando realizamos a classificação de imagens, são as únicas entradas disponíveis para o computador. A ideia é que você forneça ao computador este array de números e ele produzirá números que descrevam a probabilidade da imagem ser uma determinada classe (.98 para PARE, .02 para SIGA).  
  Agora que conhecemos o problema, bem como as entradas e saídas, vamos pensar em como abordar isso. O que queremos que o computador faça é ser capaz de diferenciar entre todas as imagens que são dadas e descobrir as características específicas que tornam uma placa de placa de pare PARE e uma placa de siga SIGA. Este processo é feito em nossas mentes subconscientemente também. Quando olhamos para uma foto de um cachorro, podemos classificá-la como tal, se a imagem tiver características identificáveis, como patas ou quatro patas. De maneira semelhante, o computador pode executar a classificação de imagens procurando por recursos de baixo nível, como bordas e curvas, e então construindo conceitos mais abstratos através de uma série de camadas convolucionais. Esta é uma visão geral do que uma CNN faz.

## Sistema de detecção YOLO

O YOLO é um sistema de detecção de objetos que utiliza uma abordagem diferente da maioria dos sistemas com esse objetivo: ele aplica uma rede neural para toda imagem, uma única vez. Essa rede retorna uma outra imagem com caixas ao redor de possíveis objetos conhecidos por essa rede já treinada. E o conjunto dessas caixas reforça a certeza na detecção desses objetos, pois quanto maior a espessura da caixa, maior a precisão na detecção. Para utilizar esse sistema, [Joseph Redmon](https://github.com/pjreddie) criou uma implementação desse [estudo](https://arxiv.org/abs/1506.02640), chamada [darknet](https://github.com/pjreddie/darknet): fácil de instalar, rodar e foi escrito em C e CUDA.

YOLOv3 prevê uma pontuação de objetividade para cada delimitação caixa usando regressão logística. Este deve ser 1 se o limite caixa antes se sobrepõe a um objeto verdade da terra por mais de qualquer outra caixa delimitadora anterior. Se a caixa delimitadora anterior não é a melhor, mas sobrepõe um objeto de verdade de mais do que alguns limiares. Assim o sistema atribui apenas uma caixa delimitadora antes de cada verdade básica objeto. Se uma caixa delimitadora anterior não for atribuída a um objeto da verdade, não incorre em perda para previsões de coordenadas ou de classes, apenas objetividade.

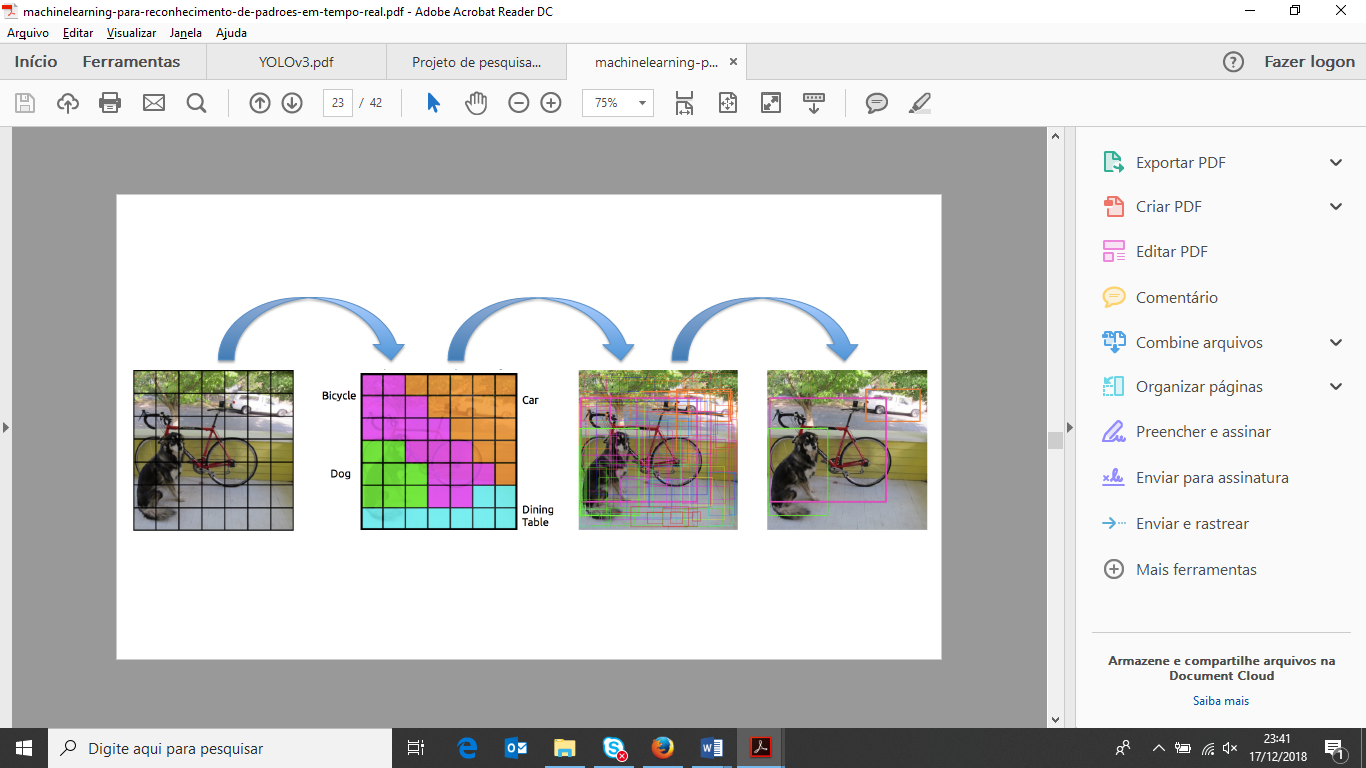


Figura IV. Identificação dos objetos detectados pela Darknet

## Veículo Autônomo IARA

Um veículo autônomo é um sistema composto por uma plataforma de hardware e uma coleção de módulos de software responsáveis por tarefas que incluem: localização, mapeamento, localização e mapeamento simultâneos (SLAM), planejamento de caminhos, planejamento de movimento, detecção de estado de luz de tráfego e decisão de alto nível. Existe um carro desenvolvido na Universidade Federal do Espírito Santo (UFES) que é totalmente autônomo chamado Automóvel Robótico Autônomo Inteligente (IARA). Foi utilizado um Ford Escape Hybrid como plataforma de hardware e foi projetado e construído um conjunto completo de módulos de software que permitem sua operação autônoma no tráfego urbano.

Existem vários métodos na literatura para abordar a problema do planejamento de movimento na estrada para carros autônomos. Bautista mostra uma revisão sobre esses métodos. Entre aqueles que foram avaliados experimentalmente usando carros autônomos do mundo real, os métodos de planejamento em estradas empregam principalmente a estrutura do estado, árvore aleatória de rápida exploração (RRT), interpolação, otimização e técnicas preditivas de modelo.

Segundo Guidolini, em métodos baseados em rede de estados, as trajetórias entre os estados inicial e final desejado são buscados em um treliça de estado adaptada para o planejamento de movimento e apenas afirma, a priori, que é provável que a solução esteja representada. As possíveis trajetórias são avaliadas por um custo-função que considera dinâmica, ambiente e restrições comportamentais, entre outros. A desvantagem destes métodos é que eles são computacionalmente pesados, devido à avaliação de cada possível solução no gráfico.

O planejador de movimento mais parecido com do IARA é o proposto por Ferguson. No entanto, este trabalho difere do de Ferguson em três aspectos principais. Primeiro, o planejador é capaz de calcular mais trajetórias complexas, uma vez que utiliza uma ranhura de ângulo de um ponto de nó extra (quatro no total) para parametrizar o carro controle, enquanto que de Ferguson usa um nó de três curvaturas. Em segundo lugar, o planejador do IARA pode gerar trajetórias otimizadas para estradas sinuosas com obstáculos, já que sua função de custo considera a estrada desejada, bem como obstáculos, enquanto o de Ferguson gera várias trajetórias alternativas e pode selecionar uma colisão. Terceiro, apresenta o algoritmo que foi usado para calcular a tabela de escolhas.

Berriel propôs um sistema que trabalhe em uma sequência temporal de imagens aplicando um mapeamento de perspectiva inversa e um filtro de partícula para cada imagem para rastrear a pista ao longo do tempo. O sistema gera uma spline cúbica para cada uma das faixas da direita e da esquerda (na pista do carro), além de cada classe de pista. Recentemente, Redes Neurais Convolucionais (CNNs) superaram o processamento de imagens com mais precisão que os outros métodos. Lee destaca que construiu um conjunto de dados com 20.000 imagens com pistas rotuladas e treinadas numa CNN para detectar e rastrear todas as faixas e marcações de estrada na imagem em uma segmentação de pixels. O sistema gera um mapa de confiança que detecta e classifica as pistas na imagem. Esses métodos, no entanto, são usados apenas na presença de marcas de estrada, não produz um mapa da estrada e não foram testados em um carro autônomo real. Além do mais, o método apresentado por Lee é fortemente dependente de parâmetros.

III. Material e métodos

## Rede YOLOv3

O YOLOv3 é extremamente rápido e preciso. Em MAP medido a 0,5 IOU, YOLOv3 está no mesmo nível da Perda Focal, mas cerca de 4 vezes mais rápido. Além disso, pode facilmente alternar entre velocidade e precisão simplesmente alterando o tamanho do modelo, sem necessidade de treinamento.

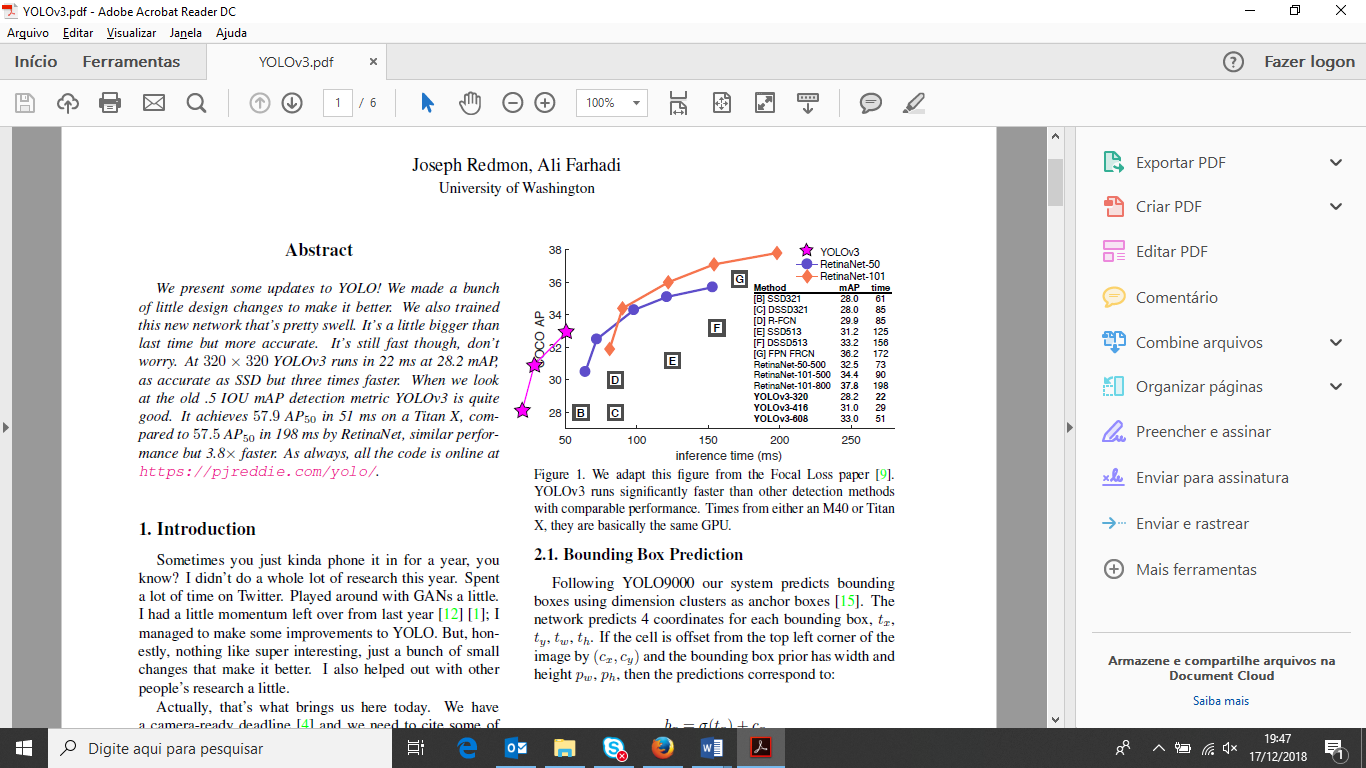


Figura V. O YOLOv3 é executado significativamente mais rápido que outros métodos de detecção com desempenho comparável.

Sistemas de detecção prévia redirecionam classificadores ou localizadores para realizar a detecção. Eles aplicam o modelo a uma imagem em vários locais e escalas. Regiões de alta pontuação da imagem são consideradas detecções. A YOLOv3 aplica uma abordagem totalmente diferente. Utiliza uma única rede neural à imagem completa. Essa rede divide a imagem em regiões e prevê caixas delimitadoras e probabilidades para cada região. Essas caixas delimitadoras são ponderadas pelas probabilidades previstas.

O modelo tem várias vantagens sobre sistemas baseados em classificadores. Ele olha para a imagem inteira em tempo de teste para que suas previsões sejam informadas pelo contexto global na imagem. Ele também faz previsões com uma única avaliação de rede, ao contrário de sistemas como R-CNN, que exigem milhares para uma única imagem. Isso faz com que seja extremamente rápido, mais de 1000 vezes mais rápido que o R-CNN e 100 vezes mais rápido que o Fast R-CNN.

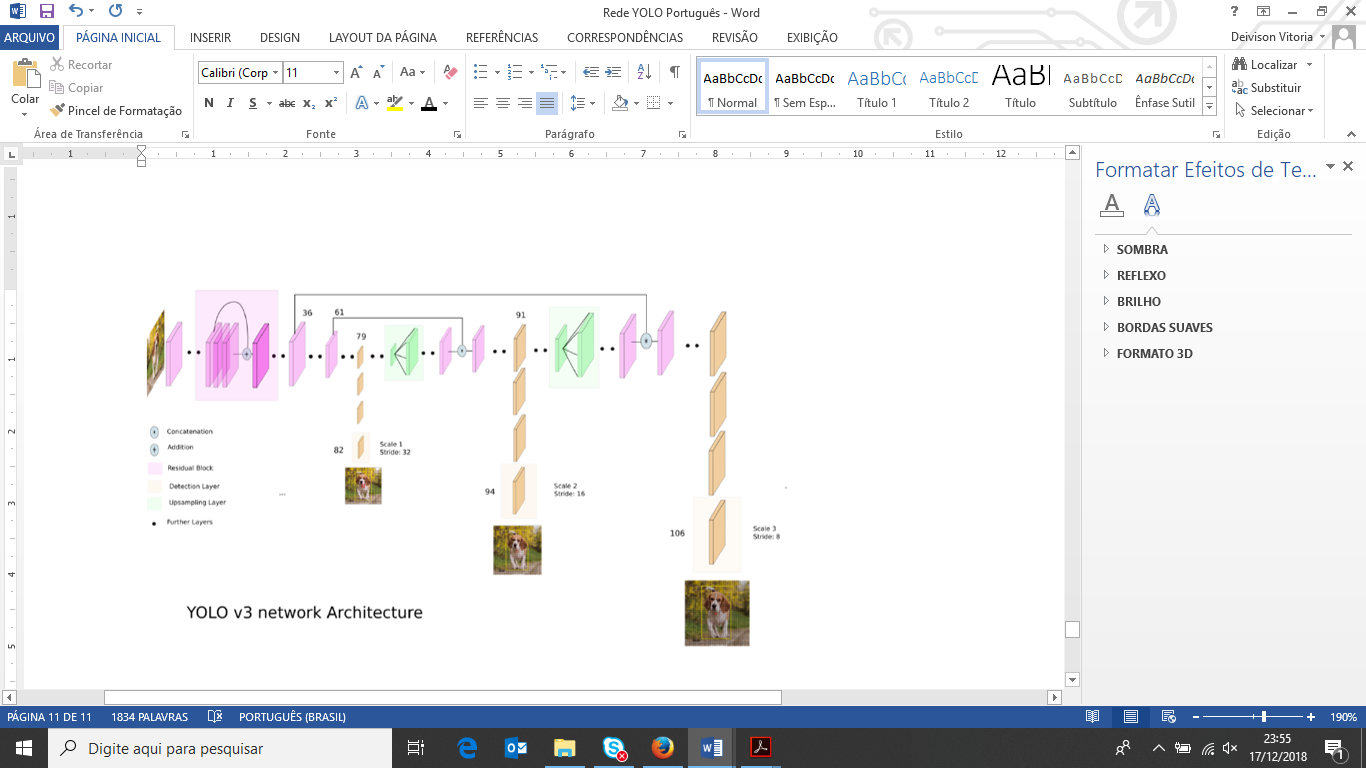


Figura VI. Arquitetura da rede YOLOv3

O YOLOv3 usa alguns truques para melhorar o treinamento e aumentar o desempenho, incluindo: previsões em várias escalas, um melhor classificador de backbone e muito mais.

Primeiro deve-se instalar o Darknet através do endereço:

Git clone <https://github.com/pjreddie/darknet>

cd darknet

make

Com o arquivo de configuração para o YOLO no subdiretório cfg /. terá que baixar o arquivo de peso pré-treinado (237 MB). Execute:

wget https://pjreddie.com/media/files/yolov3.weights

Em seguida, execute o detector desejado, como no exemplo abaixo:

./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg

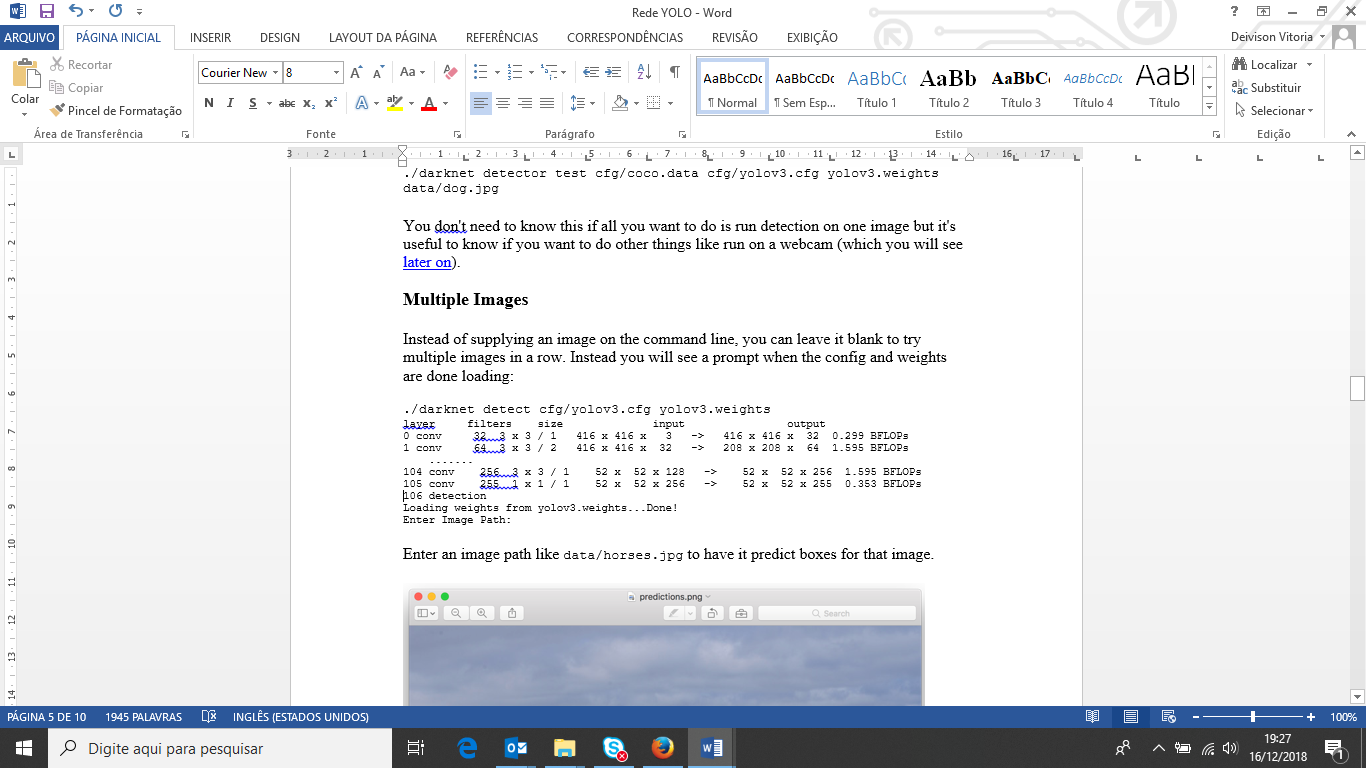
O Darknet imprime os objetos detectados, sua confiança e quanto tempo demorou para encontrá-los. Não foi compilado o Darknet com o OpenCV, então ele não pode exibir as detecções diretamente. Em vez disso, ele salva em predictions.png. Basta abri-lo para ver os objetos detectados. Se usar o Darknet na CPU, demora cerca de 6 a 12 segundos por imagem. Se usar a versão da GPU, seria muito mais rápido.

O comando detect é um atalho para uma versão mais geral do comando. É equivalente ao comando:

./darknet detector test cfg/coco.data cfg/yolov3.cfg yolov3.weights data/dog.jpg

*Imagens Múltiplas*

Em vez de fornecer uma imagem na linha de comando, podemos deixar em branco para tentar várias imagens em uma linha. Assim você verá um aviso quando a configuração e os pesos forem concluídos, carregando:



*Alterando o Limite de Detecção*

Por padrão, o YOLO exibe apenas objetos detectados com uma confiança de 0,25 ou superior. Podemos mudar isso passando o sinalizador -thresh <val> para o comando yolo. Por exemplo, para exibir toda a detecção, você pode definir o limite como 0:

./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg -thresh 0

Então, isso obviamente não é muito útil, mas pode-se configurá-lo para valores diferentes para controlar o que é atingido pelo modelo.

*Tiny YOLOv3*

Existe um modelo muito pequeno também para ambientes restritos, yolov3-tiny. Para usar este modelo, primeiro deve-se baixar os pesos:

wget https://pjreddie.com/media/files/yolov3-tiny.weights

Em seguida, execute o detector com o pequeno arquivo de configuração e pesos:

./darknet detect cfg/yolov3-tiny.cfg yolov3-tiny.weights data/dog.jpg

## Treinamento YOLOv3

## Pode-se treinar o YOLO do zero se quiser jogar com diferentes regimes de treinamento, hiper-parâmetros ou conjuntos de dados.

Para treinar a YOLOv3, utilizamos um conjunto de dados (dataset) que criamos com 4.000 dados através de imagens de PARE e SIGA em diferentes cenários. A partir disso, criamos um diretório para armazenar tudo.

Agora haverá um subdiretório VOCdevkit / com todos os dados de treinamento.

Para o treinamento, usamos pesos convolucionais que são pré-treinados na Imagenet. Nós usamos pesos do modelo darknet53. Você pode apenas baixar os pesos para as camadas convolucionais aqui (76 MB).

Wget https://pjreddie.com/media/files/darknet53.conv.74

## Aplicando classificação de imagens

Após treinar a rede fizemos a validação e conseguimos bons resultados. O treinamento girou em torno de 4 dias.

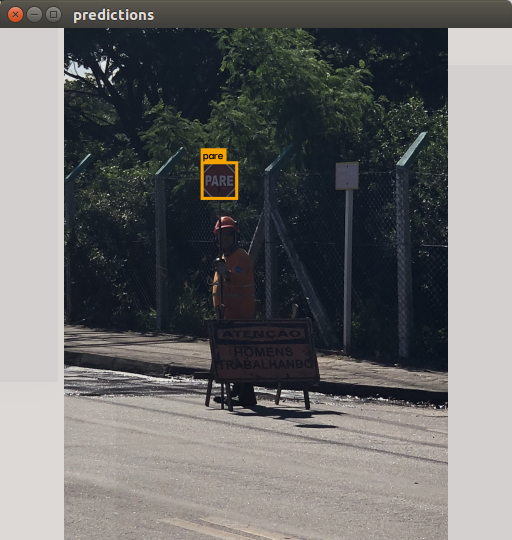
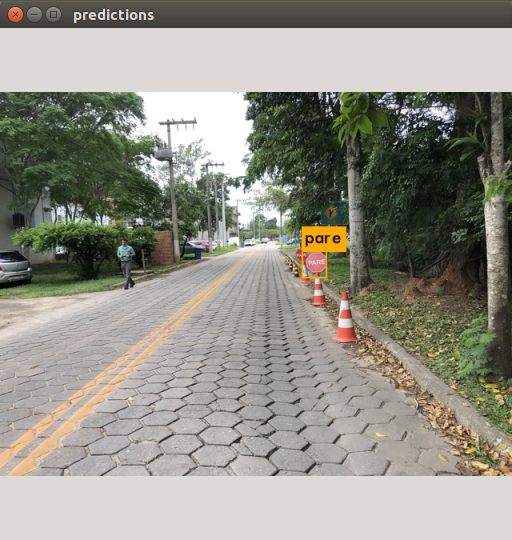
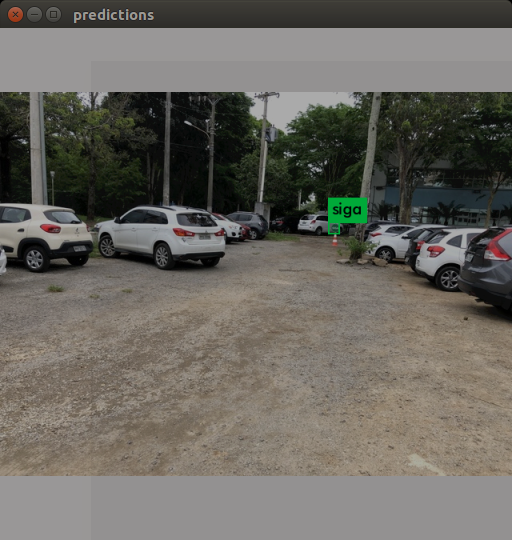
  

Figura VII. Resultados da validação: esquerda para direita:100%, 76% e 84% de acurácia.

# Experimentos e análises

O modelo introduzido na seção anterior é avaliado nos experimentos. O conjunto de dados é apresentado além das configurações de parâmetros para os experimentos são ilustradas. No final, analisamos e discutimos os resultados obtidos com os testes na rede YOLOv3.

**INSERIR OS RESULTADOS DOS TESTES REALIZADOS.**

V. Conclusão e TRABALHOS FUTUROS

O trabalho teve foco em desenvolver uma rede neural convolucional que permita que o veículo autônomo IARA consiga detectar operações de PARE e SIGA durante um trajeto numa via. Para isso, foi utilizado a rede YOLOv3 para treinar e validar a rede.

O resultado da acurácia de XXX% mostra que o resultado foi satisfatório e que podemos utilizar a rede no IARA durante suas operações.

**INSERIR UM GRÁFICO COMPARANDO O RESULTADO DA ACURRÁCIA COM AS DEMAIS REDES**

Como trabalho futuro, podemos criar um dataset com mais imagens de PARE e SIGA em cenários diferentes. Assim podemos para treinar e validar a rede de forma mais abrangente.

VI. Referências

[1] HINTON, G. E., Osindero, S., and The, Y. A fast learning algorithm for deep belief nets. Neural Computation, 18:1527{1554, 2006.

[2] HINTON, G.E., Srivastave, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580, 2012.

[3] KRIZHEVSKY, A., Sutskever, I., and Hinton, G.E. Imagenet classication with deep convolutional neural networks. In NIPS, 2012.

[4] LECUN, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. Neural Comput., 1(4):541{551, 1989.

[5] SOHN, K., Jung, D., Lee, H., and Hero III, A. Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In ICCV, 2011.

[6] D. G. Bautista, J. Pérez, V. Milanés and F. Nashashibi, “A Review of Motion Planning Techniques for Automated Vehicles”, IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1135 – 1145, 2015.

[7] D. Ferguson, T. Howard and M. Likhachev, “Motion Planning in Urban Environments”, Journal of Field Robotics, vol. 25, no. 11–12, pp. 939–960, 2008.

[8] R. F. Berriel, F. S. Rossi, A. F. de Souza, and T. Oliveira-Santos, “Automatic large-scale data acquisition via crowdsourcing for crosswalk classification: A deep learning approach,” Computers & Graphics, vol. 68, pp. 32–42, 2017.

[9] S. Lee et al., “VPGNet: Vanishing Point Guided Network for Lane and Road Marking Detection and Recognition,” In IEEE International Conference on Computer Vision, pp. 1965-1973, 2017.

[10] R. Guidolini, C. Badue, M. Berger, A. F. De Souza, "A Simple Yet Effective Obstacle Avoider For The IARA Autonomous Car", 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC 2016), 2016.

[11] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. arXiv, 2018.