FuseNet: profundidade realmente ajuda?

Leonardo Amato Loriato
leoloriato@gmail.com

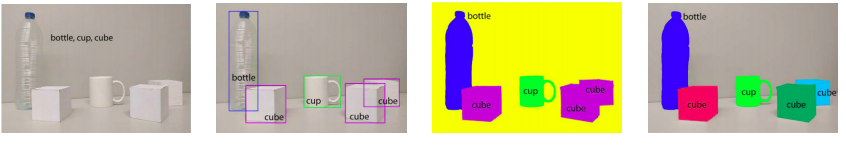
Departamento de Ciência da Computação
Universidade de Brasília
Campus Darcy Ribeiro, Asa Norte
Brasília-DF, CEP 70910-900, Brazil,

Abstract

O presente trabalho teve por objeto testar se a profundidade do dataset NYUDv2 incrementa a performance da *FuseNet* para segmentação semântica de imagens. Para se testar essa hipótese, foram treinados o modelo original proposto, uma alteração desse sem o *branch depth* e 4 alterações intermediárias. Como resultado, comprovou-se que a adição da profundidade não permite ganhos expressivos à *FuseNet* ao longo das épocas treinadas. O modelo original obteve a segunda pior IoU, ao passo que a versão sem *depth* obteve a segunda melhor performance. Ao final, propõe-se uma modificação da arquitetura, batizada de *LateFuseNet*, desligando-se os 4 primeiros blocos de fusões.

1 Introdução

Segmentação semântica de imagens é um problema da Visão Computacional que visa a determinar a exata localização dos objetos de determinadas classes em uma imagem, inserindo-se no contexto da sub-área de reconhecimento de objetos, conforme a figura 1. Antes do advento das redes neurais convolucionais (CNNs) por LeCun *et al.* [14] e de sua popularização para esse tipo de tarefa, em decorrência da AlexNet [13], esse problema era solucionado por meio de métodos não supervisionados de identificação de pontos de interesse, tais como o SIFT, proposto por Lowe [18], o SURF [19] e o *Harris Corner Detector* [6].



(a) Classificação de imagens (b) Localização de objetos (c) Segmentação semântica (d) Segmentação de imagens

Figure 1: Evolução do reconhecimento de objetos. Fonte: [8]

.

Segundo Fei Fei [7] a arquitetura das redes neurais convolucionais (CNNs) parte da premissa de que as entradas são imagens e, por essa razão, é possível aplicar certas propriedades que permitem reduzir a quantidade de parâmetros e escalar a classificação de imagens grandes de uma maneira que não é possível nas redes neurais ordinárias. Nas redes

neurais, as entradas dos neurônios são tensores tridimensionais, que representam a largura, altura e profundidade, que é composta por 3 canais, no caso de datasets RGB, ou 4 canais, no caso de datasets RGB-D, também chamados de 2.5D, por Garcia-Garcia *et al*. [8].

Um desses datasets RGB-D é o NYU Depth Dataset V2 (NYUDv2), objeto deste trabalho, que, segundo Silberman *et al*. [21], consiste de 1449 imagens RGB-D, capturadas por meio do *Microsoft Kinect device* e dividas das seguinte maneira: 795 imagens para treinamento e 654 para testes. Para Garcia-Garcia *et al*., esse dataset é famoso pela sua natureza *indoor*, que o torna útil para certas tarefas de robôs em casa.

Ainda segundo Fei-Fei, além da AlexNet e da CNN de LeCun, quatro outras arquiteturas de CNN merecem nota: a ZFNet [31]; a VGGNet [26]; a GoogLeNet [28]; e a ResNet [12], sendo essa arquitetura o estado da arte em termos de CNN segundo a autora. Dentre essas arquiteturas, destaca-se a VGGNet, também conhecida pelo acrônimo VGG-16, por possuir 16 camadas de convolução, sendo essa a base da FuseNet [10], que será descrita posteriormente. Trata-se de uma arquitetura que tem, como principal evolução em relação à sua antecessora, AlexNet, a utilização de *kernels* menores nas convoluções, que reduzem a quantidade de parâmetros e aumentam a não linearidade do modelo, o que facilita seu treinamento.

Dando sequência na literatura das CNNs, Long *et al*. [16] introduziu a arquitetura denominada *Fully Convolutional Network* (FCN), na qual as camadas *fully-conected* originais das CNNs eram substituídas por imagens de resolução baixa, contendo mapas de *features* para segmentação semântica de imagens. Uma das mais famosas FCNs é U-Net [23], originalmente concebida para segmentação semântica de imagens biomédicas.

Para Garcia-Garcia *et al*. [8], as FCNs ainda são atualmente o estado da arte no uso de Deep Learning para segmentação semântica, sendo puramente impeditiva para determinados problemas, em decorrência de diversas características intrínsecas, dentre as quais se destaca a sua invariância espacial inerente, que não leva em conta determinadas informações úteis de contexto global.

Para contornar esse problema, foi concebida uma abordagem *enconder-decoder*, em que a entrada da imagem é progressivamente reduzida por meio das convoluções do ramo *encoder*, para serem posteriormente e suavemente super-amostradas (*upsampling*) no ramo *decoder*, até se atingir a resolução da imagem original. Nesse último processo duas abordagens distintas podem ser realizadas: as operações de deconvoluções, propostas por Noh *et al*. [22] em sua arquitetura denominada DeconvNet; e as operações de *smoothed unpooling*, propostas por Badrinarayanan *et al*. [2], no âmbito da SegNet.

No contexto de datasets RGB-D, foco deste trabalho, para Hazirbas *et al*. [11], a abordagem mais trivial para lidar com segentação semântica nesse tipo de dataset seria a inserção de um quarto canal de profundidade D, normalizado de 0 a 255, em adição aos 3 canais RGB, em CNNs como a VGG-16 Net. Contudo, segundo o autor, essa abordagem é pobre, não explorando todo o potencial que o mapa de profundidade pode dar para a cena codificada na imagem.

Nesse sentido, Gupta *et al*. [10] propôs uma abordagem denominada *Horizontal Height Angle* (HHA), em que se calcula com base na disparidade D, a altura e o ângulo entre o vetor normal e o vetor gravidade, adicionando essas 3 informações ao RGB e construindo uma imagem com 6 canais de informações. Por meio dessa técnica, houve uma melhora na solução para esse problema específico de segmentação semântica RGB-D. Contudo, Hazirbas *et al*. destaca 2 problemas do HHA: o alto custo computacional para determinação dos canais HHA; e o fato dos canais RGB ainda dominarem a representação HHA, mesmo com as informações adicionais apuradas.

Para solucionar esses problemas, Hazirbas *et al*. propõe a arquitetura FuseNet, uma FCN *enconder-decoder*, inspirada na DeconvNet [22] e na SegNet [2], na qual existem dois ramos de *enconder*, além do de *decoder*: um para tratar as informações dos três canais de cores RGB das imagens e outro para tratar as informações do quarto canal D (profundidade), normalizado de 0 a 255, conforme a figura 2. Esses ramos seguem a mesma estrutura da VGG Net de 16 camadas [26], exceto pelo fato de que os blocos $fc5$, $fc6$ e $fc7$ dessa são substituídos por blocos denominados *CBR*, que consistem de uma sequência de uma normalização *batch*, uma operação de convolução e uma função de ativação *ReLU* (*Rectified Linear Unit*) [3].

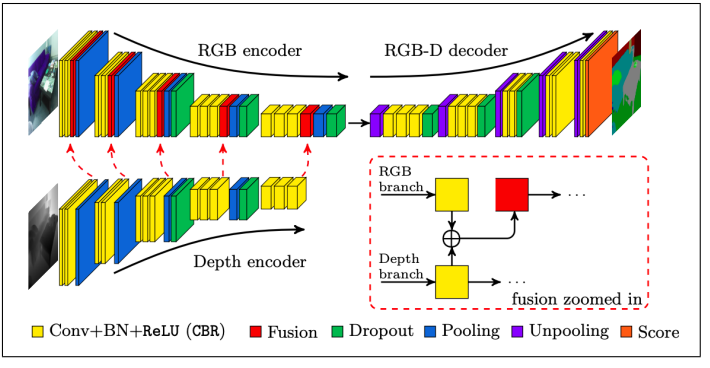


Figure 2: Arquitetura da FuseNet. Fonte: [11]

Conforme se observa, o componente-chave dessa arquitetura é o bloco de fusão, no qual os mapas de *features* provenientes das convoluções dos ramos do RGB e do D são mesclados por meio da operação de soma elemento-a-elemento nos tensores. De acordo com Hazirbas *et al*. [11], essas fusões sempre ocorrem após os blocos *CBR*, havendo porém duas abordagens possíveis de fusões: a fusão esparsa, na qual a camada de fusão é inserida apenas antes da operação de *pooling*; e a fusão densa, na qual a camada de fusão é inserida sempre após cada bloco *CBR*. Em testes realizados no *SUN RGB-D Scene Understanding Benchmark* [27], Hazirbas *et al*. verificou que ambas as estratégias possuem performance similares, havendo uma ligeira melhoria quando utilizada a estratégia de fusão esparsa.

Por fim, além desses ramos *enconder*, existem um único ramo *decoder*, no qual as informações dos mapas de *features* RGB-D são progressivamente super-amostradas (*upsample*) até atingirem o tamanho normal original da imagem com a segmentação dos objetos na imagem. Esse *upsample* é feito por meio de *unpoolings*, conforme proposto por Badrinarayanan *et al*. [2], embora Hazirbas *et al*. [11] tenha observado uma performance similar utilizando a deconvolução proposta por Noh *et al*. [22] em experimentos utilizando os datasets da ImageNet [25] e do *SUN RGB-D Scene Understanding Benchmark* [27].

2 Metodologia

2.1 Ferramentas utilizadas e carga de dados

Para a realização do presente trabalho, foi utilizado uma máquina virtual provisionada no *Google Cloud Compute Engine*, do tipo *n1-standart-8* (8 vCPUs e 30 GB de memória RAM), com um disco rígido de 100 GB e uma placa de vídeo Nvidia Tesla K80, com 11

GB de memória de vídeo. Nessa VM, foi instalado um Ubuntu 18.04 LTS, OpenCV 3.2.0 e Python 3.6.8, Pytorch, Numpy 1.16.1, h5py e diversas outras bibliotecas Python listadas no repositório do trabalho no GitHub [17].

Para realizar os testes, utilizou-se a implementação dessa rede em Pytorch da FuseNet feita por Aygün [1], que, por sua vez, é inspirada na implementação do Projeto CycleGan e do Projeto Pix-2-pix em Pytorch, feita por Zhu [32].

Para carga dos dados, utilizou-se o script disponibilizado no projeto de Aygün, denominado *datasets/create_training_set.py*, que baixa o dataset NYUDv2 provido pelo TUM Computer Vision Group [3]. Após o download dessas dados, efetua-se o mapeamento das 894 classes originais rotuladas por Silberman *et al.* [21]. Nesse ponto, diferentemente, do código original de Aygün, que mapeia as classes originais do NYUDv2 para 40 classes, neste trabalho, utilizou-se o mapeamento para 13 classes, também fornecido pelo TUM Vision.

Após a realização desse mapeamento, as imagens são formatadas no padrão 320x240 pixels, utilizando a interpolação bilinear do OpenCV [29]. Especificamente para o canal D (profundidade) da imagem, é feita uma interpolação linear, utilizando-se a máxima e mínima profundidade dos pixels da imagem em questão, de modo que o valor do canal D seja convertido numericamente de 0 a 255, a semelhança dos canais RGB da imagem. Por fim, salva-se em um arquivo .npy um dicionário contendo os canais RGB das imagens (*rgb_images*), o canal D das imagens (*depth_images*) e o *ground-truth* da segmentação das imagens (*masks*).

2.2 Experimentos realizados

A fim de verificar a influência da profundidade na performance e no treinamento da FuseNet foram concebidos 6 experimentos diferentes com modificações dessa arquitetura, a saber: *(i)* um com a arquitetura original proposta por Hazirbas *et al.* [11], a fim de servir como grupo de controle; *(ii)* um com a modificação dessa arquitetura, removendo-se a CBR 5 do *branch depth*; *(iii)* outro removendo-se as CBRs 4 e 5 do *branch depth*; *(iv)* outro removendo-se as CBRs 2 a 5 do *branch depth*; *(v)* outro removendo-se totalmente o *branch depth* da FuseNet; e *(vi)*, por fim, um último experimento mantendo-se apenas ligado bloco de fusão da CBR 5 do *branch depth*, desligando-se todas as outras fusões.

Para cada um desses experimentos, treinaram-se os modelos com a utilização de *batches* de 4 imagens cada, obtidas por amostragem aleatória, até se completar a população de 795 imagens utilizadas, em cada época. Ao todo foram treinadas 50 épocas para cada um dos experimentos mencionados supra, sendo testadados os modelos a cada 5 épocas com a população de 654 imagens separadas no NYUDv2 para testes, sendo apuradas nessas ocasiões a métrica de *Intersection over Union* (IoU) [24].

Para otimização do modelo, foi utilizado o SGD (*Stochastic Gradient Descent*), com $momentum = 0,9$, *learning-rate* $lr = 0,001$ e $weight\_decay = 0,0005$. Como função de perda, utilizou-se a *Cross Entropy Loss* [2]. Por fim, conforme recomendação trazida por Hazirbas *et al.* [11], no treinamento foram utilizadas funções de *dropout*, a fim de aprimorar o treinamento do modelo.

Por fim, especificamente para o testes do modelo, foram utilizados *batches* seriais, *i.e.* sem amostragem aleatória no dataet NYUv2, com 1 imagem cada, até atingir a poulação total de 654 imagens separadas no NYUDv2 para testes. Também diferentemente do treinamento, no caso dos testes, não foram utilizadas funções de *dropout*, que são desabilitadas durante essa etapa.

3 Resultados

Os resultados da métrica de IoU, apurados a cada 5 épocas, em um total de 50, para cada um dos 6 experimentos, são descritos na tabela 1 e no gráfico 3. Da análise da tabela e do gráfico, observa-se que a maioria dos experimentos tiveram performance muito similar, sendo que a rede que obteve mais perfomance foi aquela em que se manteve apenas ligado o bloco de fusão da CBR5 do *branch depth*.

Época	Original	S/ CBR5	S/ CBR 4-5	S/ CBR 2-5	S/ *Depth*	Só CBR 5
5	1,97%	2,92%	2,48%	4,50%	4,68%	3,29%
10	4,86%	4,31%	4,67%	6,63%	4,73%	6,64%
15	3,84%	3,45%	3,87%	4,94%	3,96%	5,50%
20	3,04%	2,79%	1,58%	3,28%	4,40%	4,45%
25	4,68%	4,43%	4,50%	4,84%	3,61%	5,56%
30	4,68%	3,74%	3,47%	4,44%	4,26%	5,23%
35	5,93%	5,50%	5,69%	5,64%	4,92%	6,63%
40	6,87%	6,41%	7,66%	7,25%	6,03%	7,87%
45	6,93%	6,83%	7,12%	7,10%	6,62%	8,82%
50	6,51%	6,12%	7,02%	7,58%	8,08%	9,47%

Table 1: IoU no treinamento das 50 épocas de cada experimento.

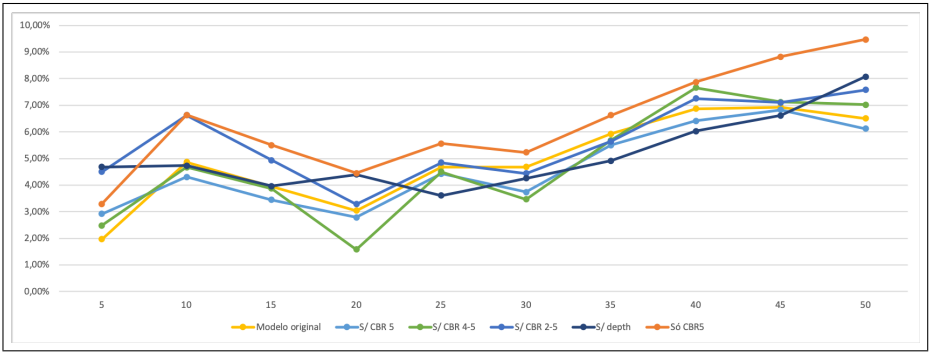


Figure 3: Evolução da IoU para cada experimento

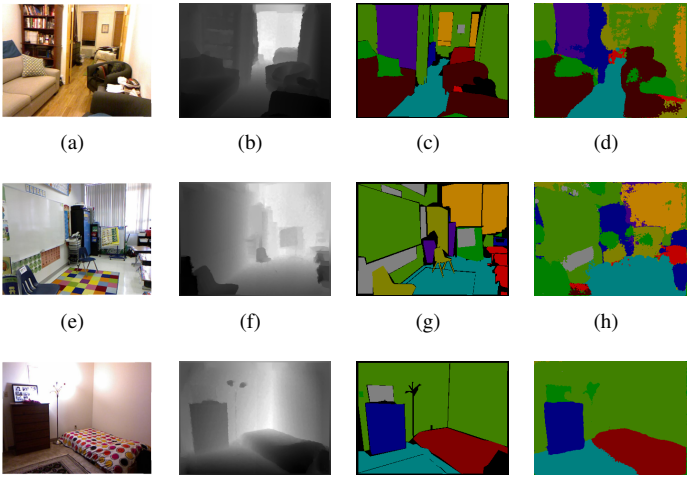
.

A segunda melhor performance foi a da rede FuseNet com o *branch depth* totalmente desligado, com resultado de IoU superior ao da rede FuseNet original. Isso, somado ao fato de que as modificações intermediárias (sem CBR5, sem CBR 4-5 e sem CBR 2-5) também obtiveram resultados similares ao da rede FuseNet original mostra que a adição do canal de profundidade não agrega muito na segmentação semântica de imagens da base RGB-D NYUDv2 por essa arquitetura. Uma explicação para isso, está no fato de, segundo Hazirbas *et al.* [11], a FuseNet ser baseada na rede VGG-16 [26] pré-treinada na ImageNet [25]. Como essa base de dados possui cerca de 14 milhões de imagens RGB rotuladas, o excesso de treinamento da rede neural nessa base acaba por suplantar qualquer aprendizado adicional a ser realizado pelo *branch* de profundidade.

Há um outro fator para o canal de profundidade não fornecer ganhos à FuseNet: segundo Couprie *et al.* [8], a segmentação semântica de objetos cuja profundidade possui alta variância na cena (*e.g.*, cama em um quarto) não performa bem no caso de uso de profundidade, sendo melhor usar apenas valores RGB para esses caso. Além disso, conforme explica Mousavian *et al.* [20], redes neurais convolucionais (CNNs) como a VGG-16, em que se baseia a FuseNet, possuem a limitação de não conseguir analisar eficientemente o contexto e as fronteiras dos objetos em uma cena durante o processo de segmentação semântica.

Isso explica também por que o modelo que mantém apenas ligado o bloco de fusão da CBR5 da FuseNet obteve a melhor performance nos experimentos, suplantando todos os demais. Em decorrência da limitação das CNNs supracitada e do fato de as primeiras convoluções de profundidade da VGG-16 serem maiores (*e.g.*, 224x224 ou 112x112), o que as torna mais ruidosas, os primeiros blocos de fusão do *branch depth* no RGB acabam por atrapalhar a convergência do modelo, sendo portanto melhor desligá-los.

Tendo a modificação da FuseNet na qual se manteve ligada apenas a fusão do CBR5, treinou-se esse modelo existoso com mais 340 épocas, em um total de 390 épocas, obtendo um total de $IoU = 40,58\%$. Os resultados dos testes desse modelo novamente treinado são discriminados na figura 4.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Figure 4: Resultados: RGB; profundidade; *ground-truth*; e saída.

.

Por fim, a tabela 2 mostra o custo computacional de treinamento dos modelos expressos em tempo médio de treinamento por época. Em que pese essa medida não possa ser avaliada em termos absolutos, uma vez que esse tempo dependa da arquitetura de computador empregada, ela pode ser avaliada em termos relativos, visto que em todos os experimentos foi-se empregado o mesmo hardware. Dessa monta, pela análise da segunda coluna, como a porcentagem de tempo que os modelos são mais rápidos em relação ao modelo original, verifica-se que esse não é nada eficiente, visto que, além de possuir perfomance similar às outras modelagens, possui um custo computacional maior.

Nada obstante, verifica-se que o tempo de treinamento médio do experimento com maior performance é praticamente o mesmo do modelo original da FuseNet. Isso é condizente com a realidade, haja vista que não foi desativada nenhuma operação de convolução do *branch depth*, que possui custo computacional bem maior que a fusão em si, que se trata de uma soma trivial elemento a elemento entre tensores.

Experimento	Tempo médio / época	% mais rápido
FuseNet Original	183 *s*	-
Sem CBR 5	175 *s*	4,57%
Sem CBR 4-5	159 *s*	15,09%
Sem CBR 2-5	135 *s*	35,56%
Sem *depth*	120 *s*	52,50%
Somente com fusão da CBR 5	182 *s*	0,55%

Table 2: Custo computacional do treinamento dos modelos em cada experimento.

4 Discussão, conclusões e trabalhos futuros

Da análise dos resultados obtidos, comprova-se a hipótese firmada de que a informação do canal de profundidade não agrega muito em termos de perfomance, medida através da métrica IoU, especificamente no modelo de CNN denomimnado FuseNet e especificamente no dataset NYUDv2. Pela IoU apurada, escolheu-se portanto o modelo com somente a fusão do CBR 5 ligada como aprimoramento da rede FuseNet, dando-lhe o nome de *LateFuseNet*, em decorrência de seguir uam abordagem *late fusion*, permitindo a fusão das convoluções do *branch depth* ao final do processo de *encoding*.

Herdando as características da VGGNet-16, a *LateFuseNet* possui a desvantagem de ser extremamente lenta para ser treinada, possuindo muitos pesos e exigindo alto consumo de memória. Dessa forma, sua utilização só é indicada quando se tem hardware e tempo suficiente para treiná-la. Caso essas premissas não sejam atendidas, sugere-se a utilização do modelo s̈em *branch depth*, que possui uma performance parecida, sendo 50% mais rápido; ou a utilização de outra abordagem existente na literatura como a RDFNet, de Seungyong *et al*. [15].

Nada obstante, a *LateFuseNet* obteve um IoU excelente para as 13 classes mapeadas no NYUDv2, diante do treinamento total de 200 épocas mostrado na seção anterior. Como trabalhos futuros, vislumbra-se a repetição dos experimentos realizados para o mapeamento de 40 classes do NYUDv2, que é comumente utilizado em diversos outros trabalhos. Também se vislumbra a repetição do experimento para um número maior de épocas, a fim de obter uma asseguração ainda mais razoável sobre a a hipótese de a profundidade não ajudar na segmentação semântica da FuseNet. Além disso, outro desdobramento seria a repetição do experimento para o dataset SUN RGB-D [27], comumente utilizado em diversos trabalhos, inclusive no trabalho original proposto por Hazirbas *et al*. [11].

Por fim, um outro trabalho futuro seria o aprimoramento da *LateFuseNet*, utilizando-se, para o *branch depth*, pesos inicializados aleatoriamente, em vez de pesos do pré-treinamento da VGG-16 [26] na ImageNet [25], como é implementada por Aygün [1]. Em que pese Yosinski *et al*. [30] afirmar que a adaptação de domínio e *transfer-learning*, por meio do *fine-tuning*, poder ser melhor que a inicialização aleatória, conjectura-se que não é o caso do *branch depth*, pois o treinamento da ImageNet, com 14 milhões de imagens, é excessivo, podendo impedir a convergência do modelo.

De todo o exposto, entende-se que os resultados alcançados no âmbito desse trabalho são excelente, comprovando-se, com asseguração razoável, que a profundidade não auxilia muito na performance da FuseNet na segmentação semântica do NYUDv2 e propondo-se uma melhoria dessa rede, batizada de LateFuseNet.

References

[1] Mehmet Aygün. Fusenet implementation in pytorch. URL https://github.com/MehmetAygun/fusenet-pytorch. Acessado em 15/11/2019.

[2] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *CoRR*, abs/1505.07293, 2015. URL http://arxiv.org/abs/1505.07293.

[3] Jason Brownlee. A gentle introduction to the rectified linear unit (relu), 2018. URL https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural- Acessado em 29/10/2019.

[4] Torch Contributors. Pytorch documentation: Torch.nn - loss functions - crossentropyloss, 2019. URL https://pytorch.org/docs/stable/nn.html#crossentropyloss. Acessado em 17/11/2019.

[5] Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information, 2013.

[6] OpenCV Documentation. Harris corner detector, 2019. URL https://docs.opencv.org/3.4/d4/d7d/tutorial_harris_detector.html.

[7] Fei Fei. Cs231n convolutional neural networks for visual recognition. URL http://cs231n.github.io/convolutional-networks/. Aula do curso da Universidade de Stanford. Acessado em 29/10/2019.

[8] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018.

[9] TUM Computer Vision Group. Guideline for using fusenet. URL https://github.com/tum-vision/fusenet/tree/master/fusenet.

[10] Saurabh Gupta, Ross B. Girshick, Pablo Arbelaez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. *CoRR*, abs/1407.5736, 2014. URL http://arxiv.org/abs/1407.5736.

[11] Caner Hazirbas, Lingni Ma, Csaba Domokos, and Daniel Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Asian conference on computer vision*, pages 213–228. Springer, 2016.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances*

in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-pdf.

[14] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. URL http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf.

[15] Seungyong Lee, Seong-Jin Park, and Ki-Sang Hong. Rdfnet: Rgb-d multi-level residual feature fusion for indoor semantic segmentation. pages 4990–4999, 10 2017. doi: 10.1109/ICCV.2017.533.

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL http://arxiv.org/abs/1411.4038.

[17] Leonardo A. Loriato. Github: unb-pvc-pd6, Dezembro 2019. URL https://github.com/leoloriato/unb-pvc-pd6.

[18] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[19] Alexander Mordvintsev and Abid K. Introduction to surf (speeded-up robust features), 2013. URL https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html.

[20] Arsalan Mousavian, Hamed Pirsiavash, and Jana Košecká. Joint semantic segmentation and depth estimation with deep convolutional networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 611–619. IEEE, 2016.

[21] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[22] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015. URL http://arxiv.org/abs/1505.04366.

[23] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. URL http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a. (Disponível em arXiv:1505.04597 [cs.CV].

[24] Adrian Rosebrock. Intersection over union (iou) for object detection, Novembro 2016. URL https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. Acessado em 17/11/2019.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. Acessado em 29/10/2019.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[27] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) oral presentation*, Junho 2015.

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

[29] OpenCV Dev Team. Opencv api reference: Geometric image transformations, novembro 2019. URL https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#resize. Acessado em 17/11/2019.

[30] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014. URL http://arxiv.org/abs/1411.1792.

[31] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013.

[32] Jun-Yan Zhu. Cyclegan and pix2pix in pytorch. URL https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix. Acessado em 15/11/2019.