Comparison of SegNet and U-Net for Semantic Segmentation

Felipe Augusto Lima Reis
PUC Minas - Pontifícia Universidade Católica de Minas Gerais
R. Walter Ianni 255 - Bloco L - Belo Horizonte, MG, Brasil
falreis@sga.pucminas.br

Abstract

Image segmentation refers to the partition of an image into a set of regions to cover it, to represent a meaningful area. Before the use of deep neural networks, the best-performing methods mostly was made using hand engineered features [7]. This paper will evaluate two Deep Neural Networks for semantic segmentation: SegNet and U-Net. These DNNs are both "fully convolutional" networks that output a result with the same size as the input. The models are trained and tested using KITTI Road Dataset. The results were also evaluated using KITTI Dataset Toolkit. The tests showed that SegNet had a better performance in image segmentation than U-Net. U-Net, however, was easier to train, smaller and provide faster results than SegNet.

1. Introduction

Image segmentation refers to the partition of an image into a set of regions to cover it, to represent meaningful areas [10]. The goal is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze [3].

Segmentation has two main objectives: the first one is to decompose the image into parts for further analysis and the second one is to perform a change of representation [10]. Also, segmentation must follow some characteristics to identify regions, as it follows:

- Regions of an image segmentation should be uniform and homogeneous with respect to some characteristic, such as gray level, color, or texture [10];
- Region interiors should be simple and without many small holes [10];
- Adjacent regions of a segmentation should have significantly different values with respect to the characteristic on which they are uniform [10];
- Boundaries of each segment should be smooth, not ragged, and should be spatially accurate [10].

Semantic pixel-wise segmentation is an active topic of research [7]. Before the use of deep neural networks, the best-performing methods mostly was made using hand engineered features [7]. The success of deep convolutional neural networks for object classification led researchers to use these techniques to learn new capabilities, such as segmentation [7].

This paper will evaluate two different Deep Neural Networks for semantic segmentation and compare the results over Kitti Road Dataset [13]. The chosen neural networks are SegNet [7] and U-Net [22].

The organization of this paper is as follows. In the next Section we show related works to this paper. Section 3 contains information about the dataset used in this work as other datasets used previously to start the project. Section 4 explains the methods used to develop this project, Section 5 describe the experiments and shows the results and Section 6 concludes this paper and gives some final considerations and ideas for future works.

2. Related Work

Our approach follows the recent successes of deep neural networks for image classification [25] [3], transfer learning [19] [31] and semantic segmentation [7] [22] [24]. Also, the paper follow classical methods for segmentation [28] [2] [16] [12], developed before the usage of neural nets.

2.1. Superpixels

Before the use of deep neural networks, the best-performing methods to segmentation mostly was made using hand engineered features [7]. Some common features used pixels as basic units of processing [30]. Superpixels are the result of perceptual grouping of pixels [30]. Superpixels produces segmentation of coarse granulation, that can be call oversegmentation [30]. Superpixel segmentation showed to be a useful pre-processing step in many computer vision applications [30].

There has been a lot of research on superpixels since the term has been established [30]. In particular, various algorithms for superpixel segmentations have been proposed.

Algorithms for generating superpixels can be broadly categorized as either graph-based or gradient ascent methods [2], as described below:

- *Graph-based algorithms*: Graph-based approaches to superpixel generation treat each pixel as a node in a graph. Edge weights between two nodes are proportional to the similarity between neighboring pixels. Superpixels are created by minimizing a cost function defined over the graph [2]. Example of this category are the *Efficient Graph-Based Image Segmentation* (EGB) algorithm [12] and the *Superpixels Lattices* [16];

- *Gradient-ascent-based algorithms*: Starting from a rough initial clustering of pixels, gradient ascent methods iteratively refine the clusters until some convergence criterion is met to form superpixels [2]. In this set, its possible to cite *watersheds* algorithms [28] and the *SLIC* algorithm [2];

2.2. Fully Convolutional Networks

Each layer of data in a convnet is a three-dimensional array of size $h \times w \times d$, where h and w are spatial dimensions, and d is the feature or channel dimension [24]. The first layer is the image, with pixel size $h \times w$, and d color channels [24]. Locations in higher layers correspond to the locations in the image they are path-connected to, which are called their receptive fields [24].

Convnets are built on translation invariance. Their basic components (convolution, pooling, and activation functions) operate on local input regions, and depend only on relative spatial coordinates [24]. While a general deep net computes a general nonlinear function, a net with only layers of this form computes a nonlinear filter, which we call a deep filter or fully convolutional network. An FCN naturally operates on an input of any size, and produces an output of corresponding (possibly resampled) spatial dimensions [24].

2.2.1 SegNet

SegNet is a deep encoder-decoder architecture for multi-class pixelwise segmentation [7]. The SegNet architecture consists of a sequence of non-linear processing layers (encoders) and a corresponding set of decoders followed by a pixel-wise classifier [7] [29]. Typically, each encoder consists of one or more convolutional layers with batch normalization and a ReLU non-linearity, followed by non-overlapping max-pooling and sub-sampling [7] [29]. The sparse encoding due to the pooling process is upsampled in the decoder using the max-pooling indices in the encoding sequence [7] [29]. Figure 1 presents the architecture of SegNet.

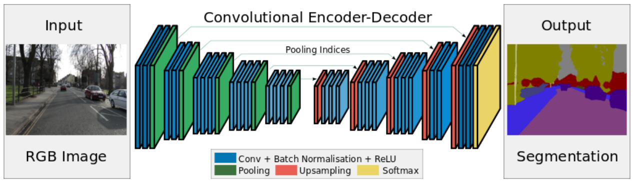

Figure 1. SegNet architecture. *Image adapted from SegNet project website* [29] [7]

2.2.2 U-Net

U-Net is a Convolutional Networks for Biomedical Image Segmentation [22] [21]. Although U-Net was developed for biomedical image segmentation, its architecture can be trained to segment other types of image.

U-Net architecture consists of the repeated application of two 3×3 convolutions, each followed by a rectified linear unit (ReLU) and a 2×2 max pooling operation with stride 2 for downsampling [22]. Every step in the expansive path consists of an upsampling of the feature map followed by a 2×2 convolution, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3×3 convolutions, each followed by a ReLU [22]. At the final layer a 1×1 convolution is used. In total the network has 23 convolutional layers [22]. Figure 2 presents the architecture of U-Net.

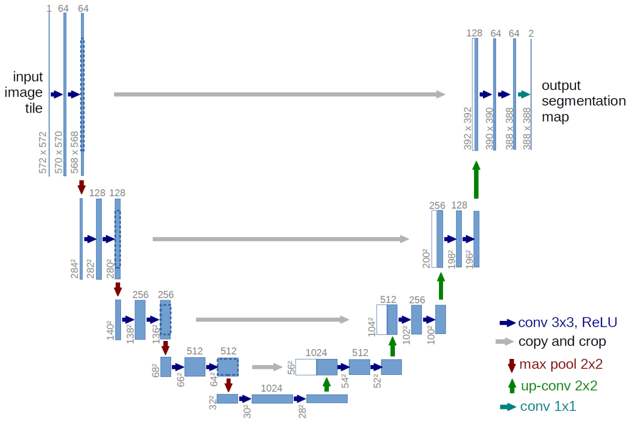

Figure 2. U-Net architecture. *Image adapted from U-Net project website* [21] [22]

3. Data

KITTI Vision Benchmarking Suite is an project of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago to provide an real-world computer vision benchmark for autonomous driving platform Annieway [14] [17]. KITTI contains benchmarks and datasets for the following area of interests: stereo, optical flow, visual odometry, 3D object detection and 3D tracking [17].

One of the benchmark in KITTI suite is the Road/Lane

Dataset Evaluation [13]. The road and lane estimation benchmark consists of 289 training and 290 test images, in four different categories of road scenes [13]:

- uu - urban unmarked (98 training images and 100 test images) [13];
- um - urban marked (95 training images and 96 test images) [13];
- umm - urban multiple marked lanes (96 training images and 94 test images) [13];
- urban - combination of the three above [13].

Ground truth has been generated by manual annotation of the images and is available for two different road terrain types: road - the road area (the composition of all lanes), and lane (the ego-lane, the lane the vehicle is currently driving on) [13] [17]. Ground truth is provided for training images only [13].

As the dataset do not provide test groundtruth, the results must be evaluated using a benchmarking tool provided with the dataset [17]. This tool performs road and lane estimation in the bird's-eye-view space [13] [17]. The metrics used are Maximum F1-measure, Average precision as used in PASCAL VOC challenges, Precision, Recall, False Positive Rate, False Negative Rate, F1 score and Hit Rate [13] [17].

3.1. Other Evaluated Datasets

The two datasets presented in this section were evaluted also for this paper, but, the final results does not contains both. The reasons to prefer KITTI over CamVid and BSDS500 datasets are available in Sections 5 and 6.

3.1.1 CamVid Dataset

The Cambridge-driving Labeled Video Database (CamVid) is the first collection of videos with object class semantic labels, complete with metadata [8]. The database provides ground truth labels that associate each pixel with one of 32 semantic classes [8]. The database addresses the need for experimental data to quantitatively evaluate emerging algorithms [8].

3.1.2 BSDS500 Dataset

Berkeley Segmentation Data Set contains 500 natural images and its respective ground-truths, annotated by humans [6]. The images are explicitly separated into disjoint train, validation and test subsets [6].

To evaluate the quality of the segmentation methods, BSDS500 provides a benchmarking tool [6]. BSDS500 dataset uses the Precision and Recall Method to evaluate the results [6].

4. Methods

4.1. Transfer Learning

Transfer learning is a technique in machine learning that stores knowledge gained while solving one problem, adapt and apply it to a different but related problem. As the growing of neural networks usage, it becomes reasonable to seek out methods that avoid "reinventing the wheel", and instead are able to build on previously trained networks' results [19] [31].

As the images of KITTI Dataset are really different from any other tested images, the usage of transfer learning needs to shrink some parts of the image and grow other, defforming a lot the original image. Then, this work started trying to use transfer learning, but decided set aside the results.

4.2. Data Augmentation

Data augmentation consists of a range of transformations that can be applied to the dataset to increase the number of data with the target of improving the accuracy and robustness of classifiers [11]. The problem with small datasets is that models trained with them do not generalize well [18].

Data augmentation also can act as a regularizer in preventing overfitting in neural networks and improve performance in imbalanced class problems [32]. According to Wong et al. [32], data augmentation is better to perform in data-space instead of feature-space, as long as label preserving transforms are known [32].

To provide data augmentation, the training images and the respective ground-truth were fliped in the left-right direction, creating a "mirror" image. This procedure created 2 times more images to the training set, as shown in Image 3.

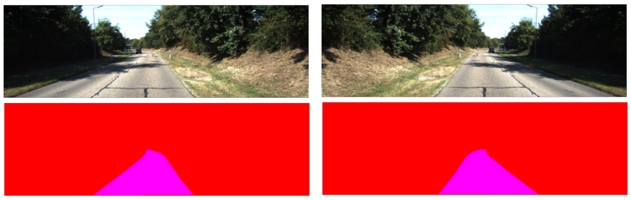

Figure 3. Data Augmentation with flip in left-right direction.

One previous procedure applied to the data was a flip in up-down direction, creating 3 more images than original dataset. The results, however, were worse than with only left-right flip, more difficult to train and slower. Up-down data-augmentation, then, was removed.

4.3. Image Reduction

KITTI Road/Lane Dataset contains a three-dimensional images with size of $375 \times 1242 \times 3$ (images size information for DNN was explained in Section 2.2). As images were too big and to speedup the training process the images were

Num.	Layer (type)	Output Shape	Parameters
1	Layer 1	(None, 3, 184, 616)	0
2	Zero Padding 2D	(None, 3, 186, 618)	0
3	Convolution 2D	(None, 64, 184, 616)	1792
4	Batch Normalization	(None, 64, 184, 616)	2464
5	Activation	(None, 64, 184, 616)	0
6	Max Pooling 2D	(None, 64, 92, 308)	0
7	Zero Padding 2D	(None, 64, 94, 310)	0
8	Convolution 2D	(None, 128, 92, 308)	73856
9	Batch Normalization	(None, 128, 92, 308)	1232
10	Activation	(None, 128, 92, 308)	0
11	Max Pooling 2D	(None, 128, 46, 154)	0
12	Zero Padding 2D	(None, 128, 48, 156)	0
13	Convolution 2D	(None, 256, 46, 154)	295168
14	Batch Normalization	(None, 256, 46, 154)	616
15	Activation	(None, 256, 46, 154)	0
16	Max Pooling 2D	(None, 256, 23, 77)	0
17	Zero Padding 2D	(None, 256, 25, 79)	0
18	Convolution 2D	(None, 512, 23, 77)	1180160
19	Batch Normalization	(None, 512, 23, 77)	308
20	Activation	(None, 512, 23, 77)	0
21	Zero Padding 2D	(None, 512, 25, 79)	0
22	Convolution 2D	(None, 512, 23, 77)	2359808
23	Batch Normalization	(None, 512, 23, 77)	308
24	Up Sampling 2D	(None, 512, 46, 154)	0
25	Zero Padding 2D	(None, 512, 46, 154)	0
26	Convolution 2D	(None, 256, 46, 154)	1179904
27	Batch Normalization	(None, 256, 46, 154)	616
28	Up Sampling 2D	(None, 256, 92, 308)	0
29	Zero Padding 2D	(None, 256, 94, 310)	0
30	Convolution 2D	(None, 128, 92, 308)	295040
31	Batch Normalization	(None, 128, 92, 308)	1232
32	Up Sampling 2D	(None, 128, 184, 616)	0
33	Zero Padding 2D	(None, 128, 186, 618)	0
34	Convolution 2D	(None, 64, 184, 616)	73792
35	Batch Normalization	(None, 64, 184, 616)	2464
36	Convolution 2D	(None, 2, 184, 616)	130
37	Reshape	(None, 2, 113344)	0
38	Permute	(None, 113344, 2)	0
39	Activation	(None, 113344, 2)	0
Total params: 5,468,890			
Trainable params: 5,464,270			
Non-trainable params: 4,620			

Table 1. SegNet layers and its number of parameters

reduced to the size of $184 \times 616 \times 3$. This size is almost half of the *height* and *width*. The size is not exactly half of the measures to keep SegNet and U-Net models with few changes, e.g. don't need to change padding, kernel sizes and other metrics, that can influence the tests.

4.4. SegNet Model

The project uses the KITTI Lane/Road dataset images reduced to the size of $184 \times 616 \times 3$, as explained in Section 4.3. The SegNet model used in this project had total of 5,468,890 parameters (5,464,270 trainable and 4,620 non-trainable). Table 1 shows the number of parameters in each layer of SegNet architecture.

As seen in Table 1, SegNet uses some techniques to improve stability and performance of it network. One of the techniques is Batch Normalization, used to simplify models saturating nonlinearities, also called as internal covariate shift [15]. Batch normalization consists in normalizing layer inputs for each training mini-batch [15]. It allows the increase of learning rates and less careful parameter initialization [15]. Batch normalization also acts as a regularizer, in some cases eliminating the need for Dropout [15], as SegNet Model.

Num.	Layer (type)	Output Shape	Parameters
1	Layer 1	(None, 3, 184, 616)	0
2	Convolution 2D	(None, 32, 184, 616)	896
3	Dropout	(None, 32, 184, 616)	0
4	Convolution 2D	(None, 32, 184, 616)	9248
5	Max Pooling 2D	(None, 32, 92, 308)	0
6	Convolution 2D	(None, 64, 92, 308)	18496
7	Dropout	(None, 64, 92, 308)	0
8	Convolution 2D	(None, 64, 92, 308)	36928
9	Max Pooling 2D	(None, 64, 46, 154)	0
10	Convolution 2D	(None, 128, 46, 154)	73856
11	Dropout	(None, 128, 46, 154)	0
12	Convolution 2D	(None, 128, 46, 154)	147584
13	Up Sampling 2D	(None, 128, 92, 308)	0
14	Concatenate (8+13)	(None, 192, 92, 308)	0
15	Convolution 2D	(None, 64, 92, 308)	110656
16	Dropout	(None, 64, 92, 308)	0
17	Convolution 2D	(None, 64, 92, 308)	36928
18	Up Sampling 2D	(None, 64, 184, 616)	0
19	Concatenate (4+18)	(None, 96, 184, 616)	0
20	Convolution 2D	(None, 32, 184, 616)	27680
21	Dropout	(None, 32, 184, 616)	0
22	Convolution 2D	(None, 32, 184, 616)	9248
23	Convolution 2D	(None, 2, 184, 616)	9248
24	Reshape	(None, 2, 113344)	0
25	Permute	(None, 113344, 2)	0
26	Activation	(None, 113344, 2)	0
Total params: 471,586			
Trainable params: 471,586			
Non-trainable params: 0			

Table 2. U-Net layers and its number of parameters

4.5. U-Net Model

The project uses the KITTI Lane/Road dataset images reduced to the size of $184 \times 616 \times 3$, as explained in Section 4.3. The SegNet model used in this project had total of 471,586 parameters (all trainable). Table 2 shows the number of parameters in each layer of SegNet architecture.

As seen in Table 2, U-Net does not contains Batch Normalization layers as SegNet. To improve stability and performance, U-Net uses Dropout Layers, a simple mecanism to avoid overfitting [26]. Also, dropout parameter don't need to be trained, reducing the number of operations.

One of important layer of U-Net, as seen in Table 2, is the Merge/Concatenate Layers.

5. Experiments

5.1. Environment

The tests described in this section were performed in a Amazon AWS [4] *g2.2xlarge* Instance. The instance contains 8 vCPUs of Intel Corporation 440FX CPU and 15GB RAM memory. Also, contains a NVIDIA Corporation GK104GL [GRID K520] GPU, with 4096 MB x2 GDDR5 memory. The custom configuration were a 100GB SSD storage capacity with a pre-configured Amazon Linux OS (Deep Learning AMI Version 12.0 - *ami-f6a94495*).

The software environment used Python 3.5 [20] language, with Keras 2.2 [9], a Python Deep Learning Library.

Keras is a high-level neural networks API of running on top of TensorFlow [1], CNTK [23], or Theano [27] [9]. In this work, Keras run over Theano [27] backend. The virtual environment was provided by Anaconda [5]. All environment requirements are available in a single file in the author's page project in Github [1].

For KITTI evaluation compatibility a new environment was created. KITTI needs Python 2.7, and older versions of NumPy and OpenCV liberaries. The conda environment requirement are available also in the author's page project in Github [2].

6. Conclusion

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, Nov 2012.

[3] S. A. Ahmed, S. Dey, and K. K. Sarma. Image texture classification using artificial neural network (ann). In *2011 2nd National Conference on Emerging Trends and Applications in Computer Science*, pages 1–4, March 2011.

[4] I. Amazon Web Services. Amazon ec2 instance types. `https://aws.amazon.com/ec2/instance-types/`, 2018.

[5] I. Anaconda. Anaconda. `https://www.anaconda.com/`, 2018.

[6] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 5 2011.

[7] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[8] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV (1)*, pages 44–57, 2008.

[9] F. Chollet et al. Keras. `https://keras.io`, 2015.

[10] D. Domnguez and R. R. Morales. *Image Segmentation: Advances*, volume 1. 2016.

[11] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard. Adaptive data augmentation for image classification. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3688–3692, Sept 2016.

[12] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, Sep 2004.

[13] J. Fritsch, T. Kuehnl, and A. Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[14] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[16] A. P. Moore, S. J. D. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[17] K. I. of Technology and T. T. I. at Chicago. The kitti vision benchmark suite. `http://www.cvlibs.net/datasets/kitti/`, 2018.

[18] L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017.

[19] L. Y. Pratt. Discriminability-based transfer between neural networks. In *Proceedings of the 5th International Conference on Neural Information Processing Systems*, NIPS'92, pages 204–211, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[20] Python.org. Python. `https://www.python.org/`, 2018.

[21] V. P. Recognition and F. o. E. Image Processing, Dept. of Computer Science. U-net: Convolutional networks for biomedical image segmentation. `https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/`, 2018.

[22] 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. (available on arXiv:1505.04597 [cs.CV]).

[23] F. Seide and A. Agarwal. Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 2135–2135, New York, NY, USA, 2016. ACM.

[24] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, April 2017.

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

[1] https://github.com/falreis/segmentation-eval/blob/master/code/i2dl.yml

[2] https://github.com/falreis/segmentation-eval/blob/master/eval/kitti.yml

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. 15:1929–1958, 06 2014.

[27] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, 5 2016.

[28] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, Jun 1991.

[29] C. Vision and U. Robotics Group at the University of Cambridge. Segnet. `http://mi.eng.cam.ac.uk/projects/segnet/`, 2018.

[30] M. Wang, X. Liu, Y. Gao, X. Ma, and N. Q. Soomro. Superpixel segmentation: A benchmark. *Signal Processing: Image Communication*, 56:28 – 39, 2017.

[31] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, May 2016.

[32] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell. Understanding data augmentation for classification: when to warp? *CoRR*, abs/1609.08764, 2016.