Segmentation of Satellite Imagery Using Fully Convolutional Networks

CS 3891

Jacob Gloudemans

Abstract

Advances in image classification using Convolutional Neural Networks have been closely followed by similar advances in image segmentation. One approach, Fully Convolutional Networks, has been found to produce good results for a variety of segmentation problems. More recent modifications of this technique have been found to produce even better results, however, the original technique is still able to achieve near-state-of-the-art performance. In this work, we apply this approach to satellite imagery, attempting to train a model that can accurately label pixels in a satellite image as belonging to cultivated land or not. Utilizing transfer learning and training the model with GPU acceleration in the Google Colaboratory environment, we are able to correctly label pixels with over 93% accuracy.

Introduction

In computer vision applications, image segmentation is the process of dividing an image into a number of classes. Whereas in traditional classification, an algorithm may simply identify which objects are present in an image, or in some cases, place bounding boxes around those objects, image segmentation involves labeling each pixel of the image according to its class. Thus, the output of an image segmentation algorithm is itself an image, ideally with the same resolution as the original image.

State-of the art image segmentation is currently being achieved by methods which use convolutional neural networks to extract feature information from the image. The first highly successful approach of this kind uses models called Fully Convolutional Networks to perform segmentation [1]. While variations on this approach have since led to improvements over the original model [2, 3, 4], pure FCNs are still highly successful and are capable of results that are near the current state-of-the-art. FCNs are the approach used for this project.

When convolutional neural networks are used for image classification, architectures are generally composed of a sequence of convolution layers (often interspersed with pooling layers) , followed by a flattening layer and a set of fully connected dense layers. The convolutional layers function as feature extractors while the dense layers use those features to identify the image class. Importantly, the output of the convolution layers is flattened before being passed into the dense layers. While doing so retains information about *which* features are present in the image, it eliminates information about *where* in the image those features are. This is fine for classification problems for which the *what* is important and the *where* is not, however, it is important for segmentation which requires knowledge of both the *what* and the *where*.

Fully convolutional networks (FCNs) overcome this issue by, as the name suggests, eliminating the flattening layer and dense layers from the typical architecture and replacing them with additional convolution layers which use 1x1 kernels. The final convolution layer uses a number of filters equal to the number of classes expected in the output image. Doing so produces a ‘heatmap’ for each class, essentially a low-resolution grid where each cell represents that probability that the corresponding in the region in the input image contains the higher class. [1]

To produce an output with the same resolution as the input image, these ‘heatmaps’ are progressively upsampled using transposed convolution layers. After each upsampling, a skip connection is added from one of the prior pooling layers, allowing for increased accuracy in the increased resolution outputs. Essentially, the network consists of an encoder (which extracts increasingly abstract class information from the image) and a decoder (which works backwards, using information from the

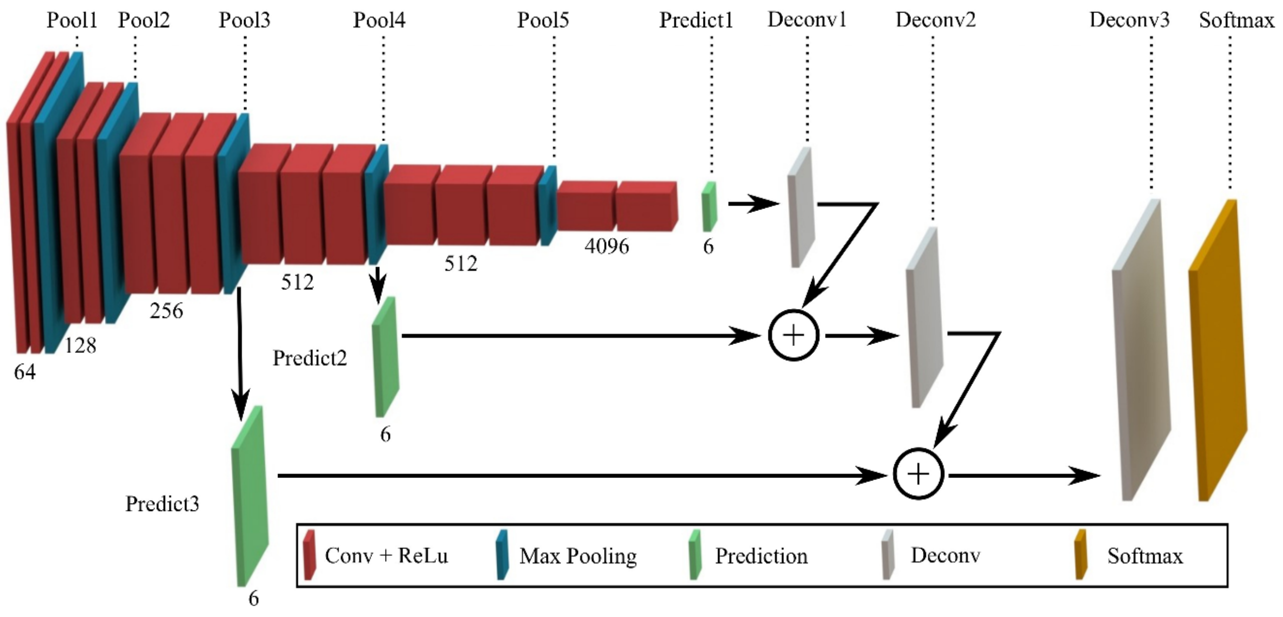

Figure 1 Fully convolutional network architecture. Note that for this project a sigmoid output activation was used rather than a softmax, and the depth of the "Predict" layers was 1 rather than 6. *Image from [5]*

encoder to increase the resolution of the output). The full architecture is shown in **Figure 1**. [1]

Note that there are slight differences between the architecture shown in the figure and the one used for this project. Because our output only has two classes, the final softmax activation is replaced with a sigmoid activation, and the dimension of the output is 1 x 224 x 224.

In this project, we attempt to train an FCN to segment satellite images into two classes: cultivated land and uncultivated land. Ground truth images were obtained from the USDA NASS Cropland Data Layer [6]. This dataset has two bands, each of which spans the entire surface of the United States. The first band divides land into 255 classes which include numerous types of forest, cropland, and water. The second band labels land as being cultivated or not and was used to create the ground truth outputs for this project.

While this specific task may have limited utility, it is easy to imagine how, if successful, the same approach could be used to perform a variety of interesting transformations on satellite imagery. To provide one example, this approach could be used to identify regions where deforestation has occurred, potentially helping to pinpoint and stop illegal logging or identify problematic forest loss trends.

Methodology

Data Collection

Input images were obtained from satellite imagery produced by the Sentinel-2 mission, accessed via the Google Earth Engine API [7]. Output images were obtained from the USDA NASS Cropland Data Layers collection via the same API [6]. To build the dataset, approximately 15,000 random coordinates from within the United States were generated. For each coordinate, a 3.75 km x 3.75 km square region of imagery was retrieved from each of the two sources and converted to a standard image format. 20% of the full dataset was then set aside to be used as a test set.

Building the Model

The code was developed with Python [8] and Tensorflow [9], using the Keras Functional API [10]. The neural network was developed and trained using Google Colaboratory [11], a free online development environment which provides easy access to powerful computational resources which greatly increase the speed at which the model can be trained. All code used for this project is available at this project's GitHub repository[1]:

As was recommended in [1], the successful VGG16 architecture [12] was used for the encoder portion of the model. Rather than training the entire model from scratch, we used transfer learning, initializing the network weights to those of a VGG16 model pretrained on the ImageNet dataset [13]. This speeds the training process as the early layers in the network are already at reasonable values.

Training the Model

The model was trained using minibatches of size 32, with the Adam optimizer. As each output pixel had two possible classes, pixelwise binary cross entropy was used for the loss function. 20-30 training epochs were typically sufficient for the model to reach its maximum accuracy.

Weight initialization (for the non-pretrained layers) was found to be very important in ensuring that the reached a high accuracy. We randomly initialized weights from a truncated normal distribution with a standard deviation of 0.01.

Two regularization approaches were attempted, dropout and L1 regularization. While overfitting was not a major problem during most of training, models did tend to overfit once accuracies were above roughly 90%. Both methods decreased the degree of overfitting, however, L1 was found to be more effective.

Results

Quantitatively, the model was able to achieve pixel-wise accuracy on the test set of roughly 93%[2], using the model that performed best on the validation data after training. Learning curves are shown in **Figure 2** and **Figure 3**.

Qualitatively, as is shown in **Figure 4**, the network outputs generally resemble the ground truth image quite closely. Relative to the ground truth images, the network outputs tend to have edges that aren't as sharp. Additionally the network misses some patches and occasionally predicts a patch were there is none. However, much of the cultivated land is correctly labeled as such, especially the land that is clearly farmland

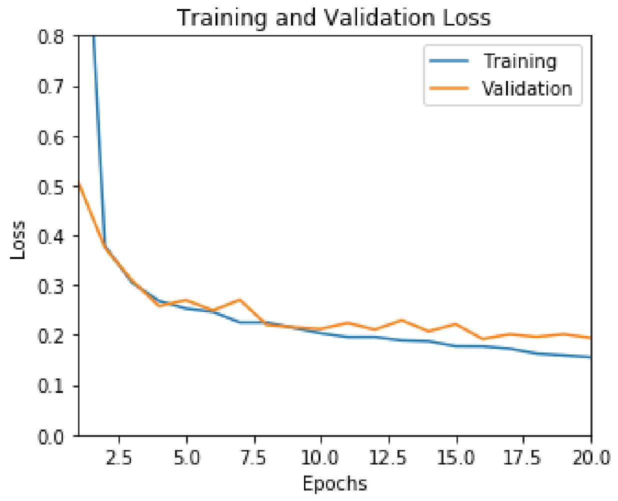

Figure 3 Training and validation loss history over 20 training epochs

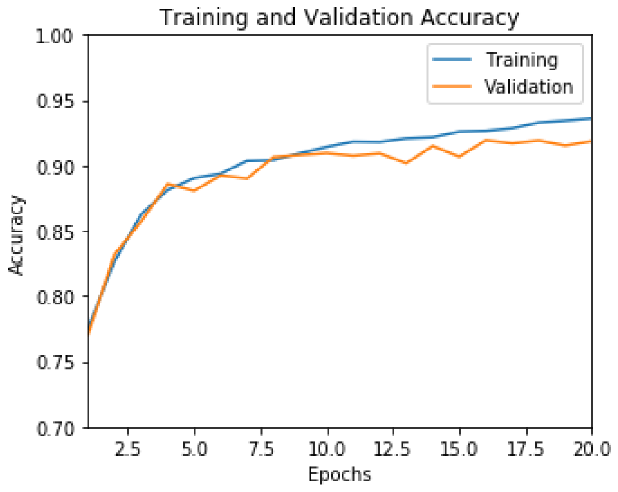

Figure 4 Training and validation accuracy history over 20 training epochs

[1] https://github.com/partlygloudy/fcn-cropland-segmentation
[2] A baseline accuracy of 78% would be achieved by labelling all pixels as "uncultivated"

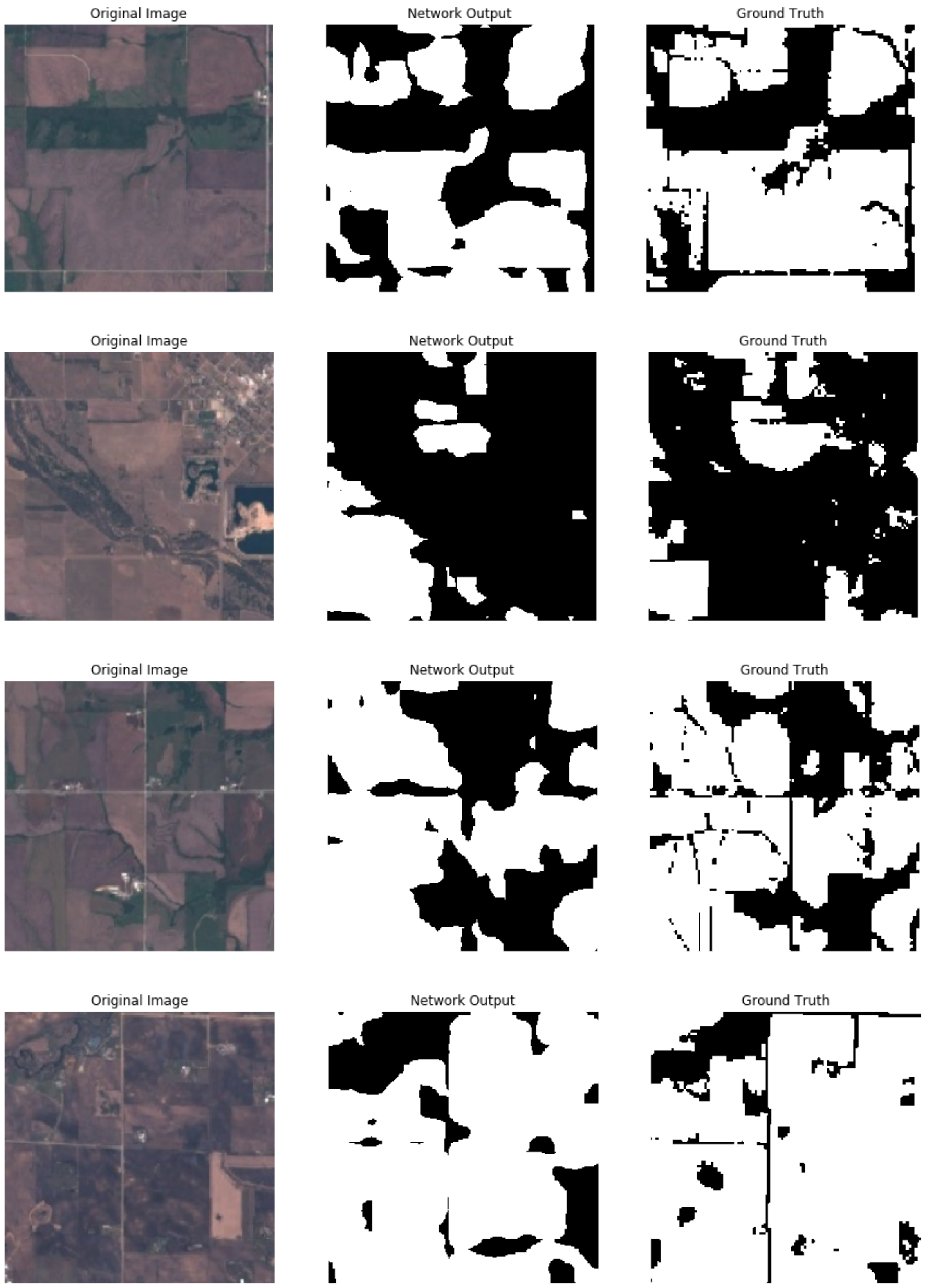

Figure 5 Inputs outputs and ground truth image examples. Left and right images were used to train the network. Middle images show network outputs. Each of these sets comes from the test set.

Discussion

Using an FCN, we were able to perform segmentation of satellite images into cultivated and uncultivated land to a high degree of accuracy. While there certainly was room for improvement, the results achieved were good enough to be useful and demonstrated that this technique is practical to use for similar applications.

A variety of steps could be taken to improve the accuracy of the model. In this case, the simplest step would be to increase the size of the dataset. Due to the limited time for this project, only 15,000 images could be downloaded, however, there is no reason that thousands of additional images couldn't be downloaded. Another step that could be taken would be to introduce noise to the input images or introduce small random affine transformations to the images. This should produce an effect similar to increasing the size of the dataset and would require no extra time as increasing the dataset size does. Finally, recent works have proposed modifications to the original FCN algorithm which have been shown to outperform the original architecture [2, 3, 4]. Implementing one of these modifications would likely lead to improvements in model performance as well.

Finally, it should be noted that the dataset being used to produce the ground truth images for training is not perfect. Briefly scanning the ground truth images, the keen observer will quickly realize that the dataset has small patches in many images which are mislabeled in one direction or the other (see **Figure 5**). While this affects only a small number of pixels, it may still amount to 1 or 2% of the total. This limits the maximum achievable accuracy that can reasonably be obtained with respect to the ground truth images.

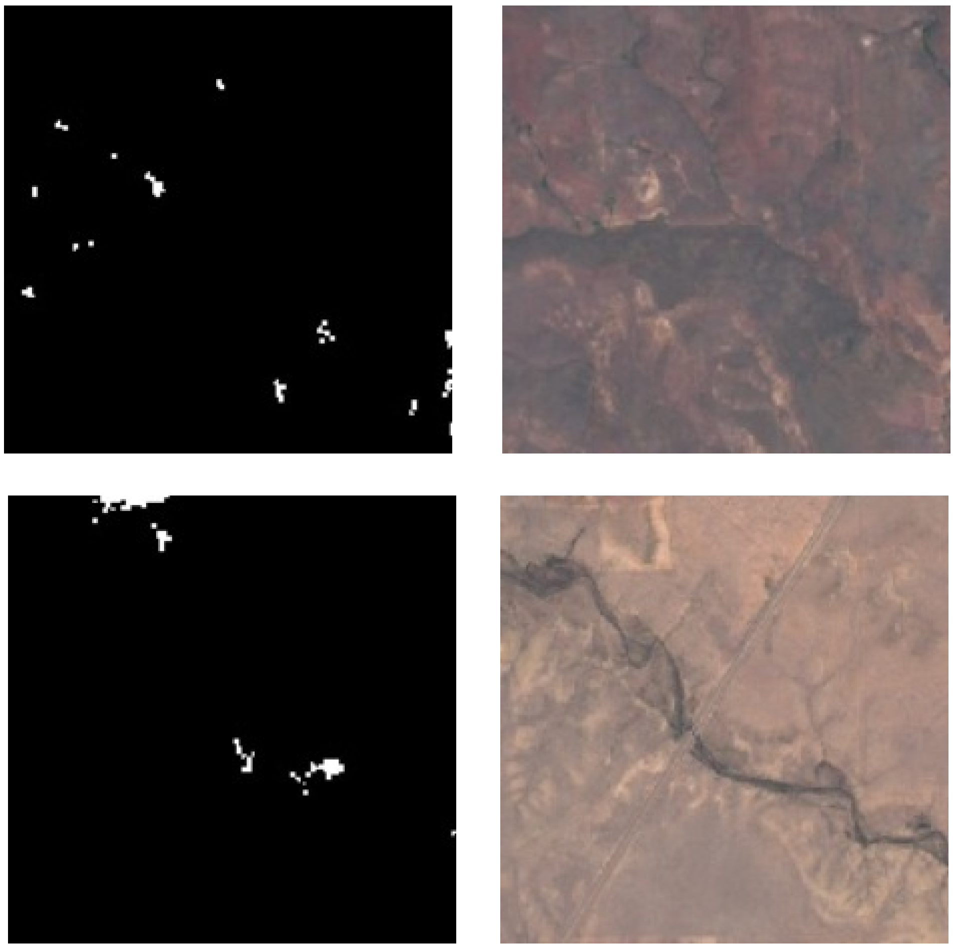

Figure 5 Some of the images in the USDA NASS dataset appear to have some erroneously classified regions. The images above show regions that are labeled as cultivated land that do not appear to actually be so.

References

[1] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 2015, pp. 3431-3440.

[2] V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481-2495, 1 Dec. 2017.

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834-848, 1 April 2018.

[4] Yu, Fisher & Koltun, Vladlen. (2015). Multi-Scale Context Aggregation by Dilated Convolutions.

[5] PCA-aided Fully Convolutional Networks for Semantic Segmentation of Multi-channel fMRI - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-structure-of-our-proposed-fully-convolutional-neural-network-Feature-learning-part_fig4_30891-2864 [accessed 26 Apr, 2019]

[6] USDA National Agricultural Statistics Service Cropland Data Layer. 2017. Published crop-specific data layer [Online]. Available at https://nassgeodata.gmu.edu/CropScape/ (accessed Apr 10, 2019; verified Apr 10, 2019). USDA-NASS, Washington, DC.

[7] Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*.

[8] Python Software Foundation, https://www.python.org/

[9] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[10] Chollet, F. (2015) keras, GitHub. https://github.com/fchollet/keras

[11] Colaboratory: Frequently Asked Questions, Jun. 2018, [online] Available: https://research.google.com/colaboratory/faq.html.

[12] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Kuala Lumpur, 2015, pp. 730-734.

[13] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, 2009, pp. 248-255.

[14] Le, James. "How to do Semantic Segmentation using Deep learning", May 3, 2018, https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef.

[15] Yumi. “Learn about Fully Convolutional Networks for semantic segmentation.”, May 25, 2018, https://fairyonice.github.io/Learn-about-Fully-Convolutional-Networks-for-semantic-segmentation.html