Image Segemantation

-Manoj Kumar Gootam

1. Image Segmentation

Image Segmentation is the process of partitioning an image into non-intersecting regions such that each region is homogenous and union of no two adjacent regions is homogenous. In simple words, it is a pixel-wise classification problem and is a natural extension of image classification.

Convolutional networks are driving advances in recognition. Convnets are not only improving for whole-image classification, but also making progress on local tasks with structured output. These include advances in bounding box object detection, part and keypoint prediction and local correspondence. The natural next step in the progression from coarse to fine inference is to make a prediction at every pixel. Prior approaches have used convnets for semantic segmentation, in which each pixel is labeled with the class of its enclosing object or region, but with shortcomings that this work addresses

2. Challenges with Image Segmentation

There are a few challenges with Image Segmentation. It is not straightforward as Image Classification because Image Segmentation deals with the Pixelwise Classification. For example from the picture below on the left side, a very good image classifier would probably say that the picture has a CAT in it. Similarly, the picture on the right side would be classified as a DOG. What if the picture has both cat and dog in it as shown in the image below. The classifier fails to identify both.

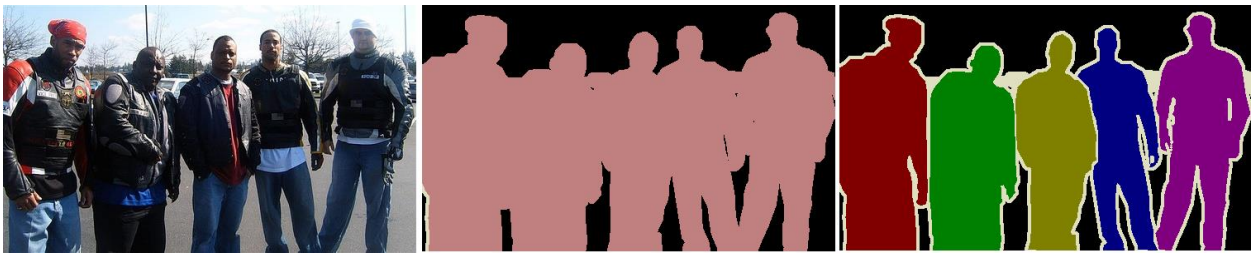
Image Segemantation

-Manoj Kumar Gootam

Image Segemantation

-Manoj Kumar Gootam

Identification of the locations of the objects is a complex task. Image classification algorithms fail when there are several instances of the same class in an image, which is called the Instance Aware Image Segmentation. Like in the picture below on the left side below there are multiple people, but the picture in the picture does not identify all the people, but the picture on the right identifies all of them. That would be ideal because there are always images with multiple instances of the same object.



Another challenging aspect of image segmentation is the computational complexity of the network architectures which usually have 100 million parameters to learn. Because of these reasons, it is difficult to come up with simple architectures with minimal computational complexity. This exercise addresses this issue by inducing a pooling mask layer which stores the max pooling indices during forward pass downsampling and upsamples them to form sparse feature maps.

3. Literature Review

There are several approaches that can be used for Image Segmentation. There are several Machine Learning based approaches that could be used like Graph Based Segmentation, Conditional Random Fields, SuperParsing - NonParametric Approach. The problem with the machine learning approaches are that they are limited in terms of computational complexity and their scalability when applied to complicated datasets. On the other hand, there are several Deep Learning based approaches like Segnet, DIGITS 5 – NVIDIA, DeepMask+SharpMask+MultiPathNet – Facebook and ENet which are computationally complicated but highly scalable.

4. Image Segmentation Approach

Semantic segmentation has a wide array of applications ranging from scene understanding, inferring support-relationships among objects to autonomous driving. Some of the applications include Medical Imaging - Diagnostic Analysis, Object Detection - Face Recognition, Iris recognition, Autonomous Driving, Image Deblurring, Image Captioning, Traffic Control Systems, Content Based Image Retrieval, Military applications. Early methods that relied on low-level vision cues have fast been superseded by popular machine learning algorithms. In particular, deep learning has seen huge success lately in handwritten digit recognition, speech, categorizing whole images and detecting objects in images. Now there is an active interest for semantic pixel-wise labelling. However, some of these recent approaches have tried to directly adopt deep architectures designed for category prediction to pixel-wise labelling. The results, although very encouraging, appear coarse. This is primarily because max pooling and sub-sampling reduce feature map resolution. Our motivation to design SegNet arises from this need to map low resolution features to input resolution for pixel-wise classification. This mapping must produce features which are useful for

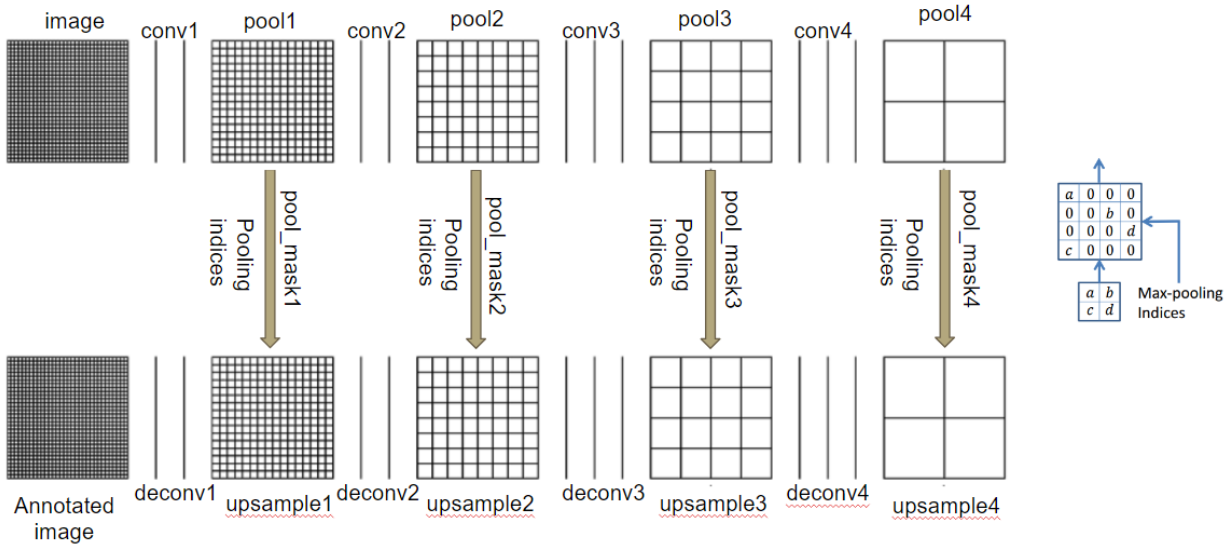
accurate boundary localization. Our architecture, SegNet, is designed to be an efficient architecture for pixel-wise semantic segmentation. It is primarily motivated by road scene understanding applications which require the ability to model appearance (road, building), shape (cars, pedestrians) and understand the spatial-relationship (context) between different classes such as road and side-walk. In typical road scenes, the majority of the pixels belong to large classes such as road, building and hence the network must produce smooth segmentations. The engine must also have the ability to delineate objects based on their shape despite their small size. Hence it is important to retain boundary information in the extracted image representation. From a computational perspective, it is necessary for the network to be efficient in terms of both memory and computation time during inference. The ability to train end-to-end in order to jointly optimise all the weights in the network using an efficient weight update technique such as stochastic gradient descent (SGD) is an additional benefit since it is more easily repeatable. The design of SegNet arose from a need to match these criteria.

The encoder network in SegNet is topologically identical to the convolutional layers in VGG16 . We remove the fully connected layers of VGG16 which makes the SegNet encoder network significantly smaller and easier to train than many other recent architectures . The key component of SegNet is the decoder network which consists of a hierarchy of decoders one corresponding to each encoder. Of these, the appropriate decoders use the max-pooling indices received from the corresponding encoder to perform non-linear upsampling of their input feature maps. This idea was inspired from an architecture designed for unsupervised feature learning. Reusing max-pooling indices in the decoding process has several practical advantages;

(i) it improves boundary delineation ,

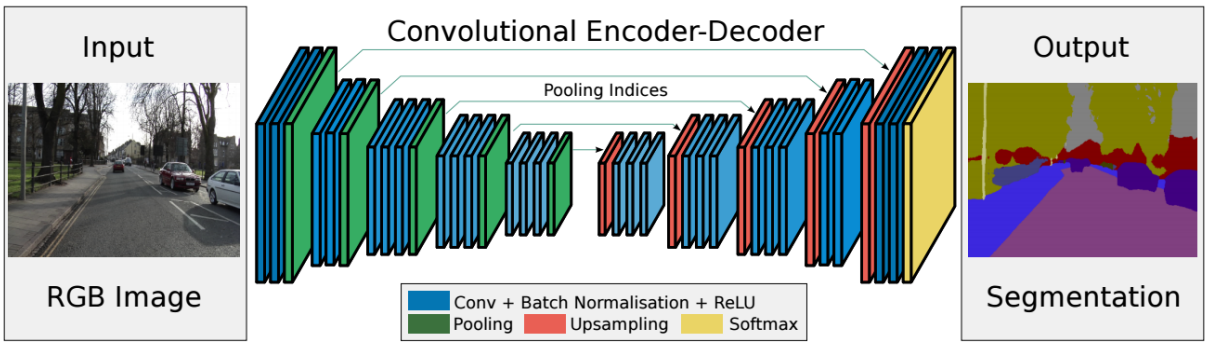
(ii) it reduces the number of parameters enabling end-to-end training, and

(iii) this form of upsampling can be incorporated into any encoder-decoder architecture with only a little modification.



From the above picture, we can try to understand how SegNet works. It takes an image and is operated upon by several consecutive convolutional layers that include convolution, pooling, batch-normalization and max pooling layers. This portion of the network architecture is called an encoder. Then there are

decoder convolutional networks corresponding to each of the encoding layers which upsample based on the max pooling indices from the encoder network and generate sparse feature maps.



SegNet is a caffe based Deep Learning Library used for convolutional nets with state of the art algorithms built on top of Caffe. This includes several new layers like DenseImageData, Pooling Mask Layers and uses several layers from Caffe like Batch Normalization, ReLU, Convolution, Max Pooling, Soft Max With Loss

SegNet has an encoder network and a corresponding decoder network, followed by a final pixelwise classification layer. This architecture is illustrated in the above figure. The encoder network consists
Of 13 convolutional layers which correspond to the first 13 convolutional layers in the VGG16 network designed for object classification. We can therefore initialize the training process from weights trained for classification on large datasets. We can also discard the fully connected layers in favour of retaining higher resolution feature maps at the deepest encoder output. This also reduces the number of parameters in the SegNet encoder network significantly (from 134M to 14.7M) as compared to other recent architectures. Each encoder layer has a corresponding decoder layer and hence the decoder network has 13 layers. The final decoder output is fed to a multi-class soft-max classifier to produce class probabilities for each pixel independently.
Each encoder in the encoder network performs convolution with a filter bank to produce a set of feature maps. These are then batch normalized. Then an element-wise rectified-linear non-linearity (ReLU) max (0, x) is applied. Following that, max-pooling with a 2×2 window and stride 2 (non-overlapping window) is performed and the resulting output is sub-sampled by a factor of 2. Max-pooling is used to achieve translation invariance over small spatial shifts in the input image. Sub-sampling results in a large input image context (spatial window) for each pixel in the feature map. While several layers of max-pooling and sub-sampling can achieve more translation invariance for robust classification correspondingly there is a loss of spatial resolution of the feature maps. The increasingly lossy (boundary detail) image representation is not beneficial for segmentation where boundary delineation is vital. Therefore, it is necessary to capture and store boundary information in the encoder feature maps before sub-sampling is performed. If memory during inference is not constrained, then all the encoder feature maps (after sub-sampling) can be stored. This is usually not the case in practical applications and hence we propose a more efficient way to store this information. It involves storing only the max-pooling indices, i.e, the locations of the maximum feature value in each pooling window is memorized for each encoder feature map. In principle,
this can be done using 2 bits for each 2×2 pooling window and is thus much more efficient to store as compared to memorizing feature map(s) in float precision. As we show later in this work, this lower memory storage results in a slight loss of accuracy but is still suitable for practical applications.

The appropriate decoder in the decoder network upsamples its input feature map(s) using the memorized max-pooling indices from the corresponding encoder feature map(s). This step produces sparse feature maps. This SegNet decoding technique is illustrated in the above figure. These feature maps are then convolved with a trainable decoder filter bank to produce dense feature maps.
A batch normalization step is then applied to each of these maps. Note that the decoder corresponding to the first encoder (closest to the input image) produces a multi-channel feature map, although its encoder input has 3 channels (RGB). This is unlike the other decoders in the network which produce feature maps with the same number of size and channels as their encoder inputs. The high dimensional feature representation at the output of the final decoder is fed to a trainable soft-max classifier. This soft-max classifies each pixel independently. The output of the soft-max classifier is a K channel image of probabilities where K is the number of classes. The predicted segmentation corresponds to the class with maximum probability at each pixel.

5. Experiments on synthetically generated dataset with 2 classes, square and background

In this exercise, several images have been generated with 2 classes, a square box and the background. The squares are of size mxm where m is randomly selected from 90 to 100 and the location of the center is random too. There are a total of 100 images each for Training, Testing and Validation. For each of these images, labelled/annotated images are generated with labels of 2 classes - Square and Background. In the following picture, the left one is the original input image and the right side is the annotated image of the original. The RGB colour intensity values are (0,0,0) for the background and (1,1,1) for the square object, this is the reason why the annotated image looks dark.

In this exercise a new simplified version of SegNet has been developed with the following design

- 4 Convolutional Layers (2 Encoders + 2 Decoders)
- 2 Max Pooling (downsampling) Layers
- 2 Upsampling Layers, 1 “softmaxwithloss” and 1 accuracy

Image Segemantation

-Manoj Kumar Gootam

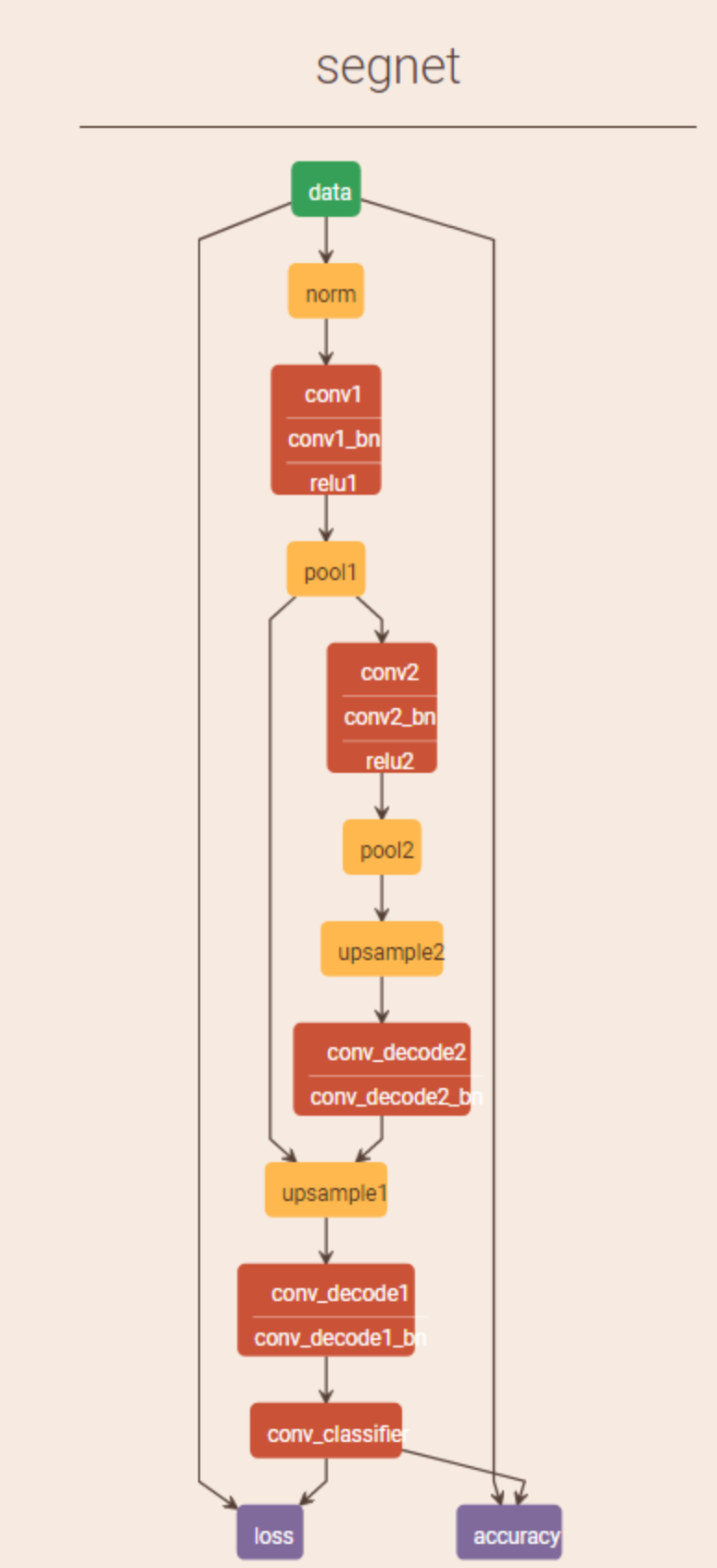


Image Segemantation

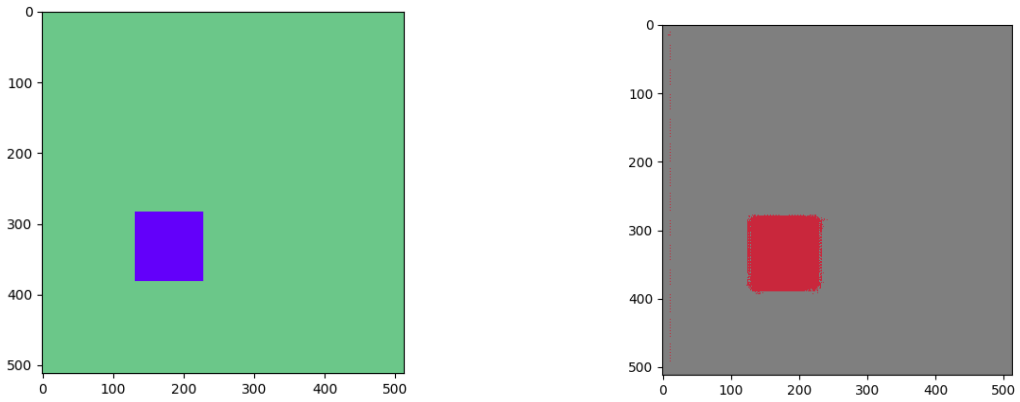
-Manoj Kumar Gootam

Approach: Trained the above architecture on the train dataset to generate the trained .caffemodel and used validation images to stop the training phase until convergence. Since, Batch Normalization shifts the feature maps based on the mean and variance of each batch during Training, we must use the same statistics for the entire dataset during testing. Test Weights are generated and used to segment the test images

Image Segmentation with 2 classes

Identifying a square object - Class 1

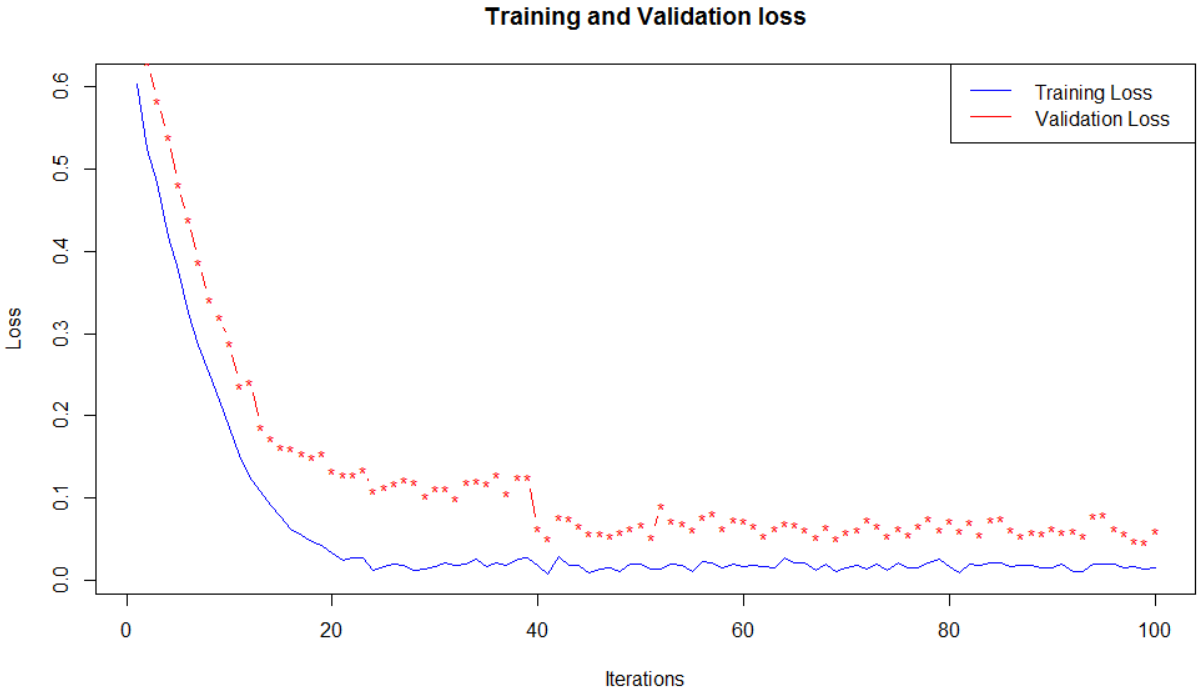
Identifying the background - Class 0

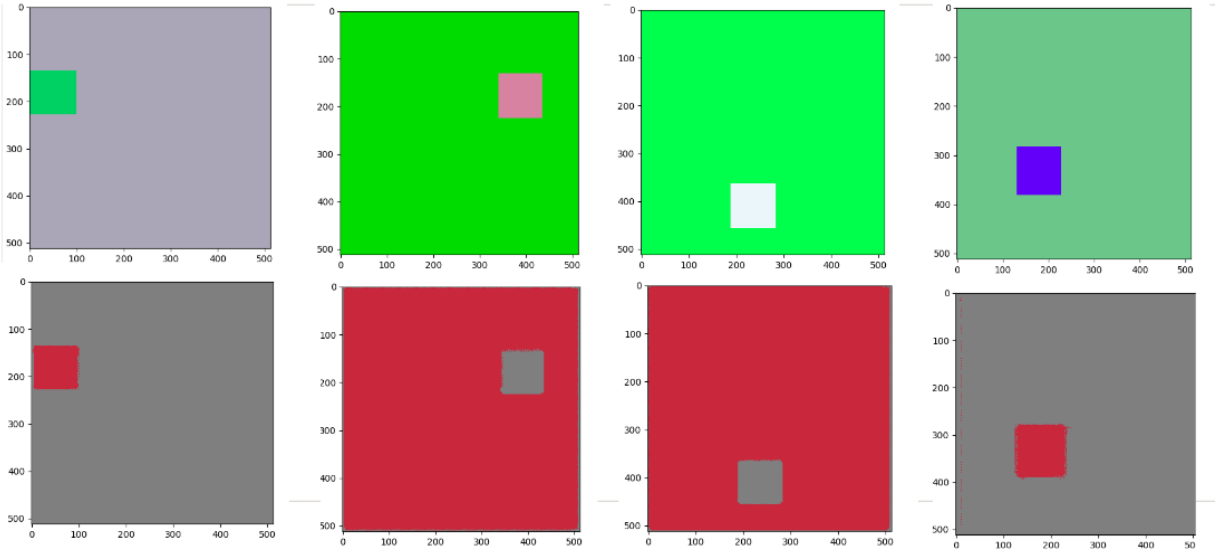


The picture on the left side of the image is the input image and the picture on the right side is the annotated image. It looks like the image segmentation is doing a good job with some loss in information near the boundaries. The following is the plot of training and validation loss. From the plot, it is clear that the algorithm converges after 40 iterations.

Image Segemantation

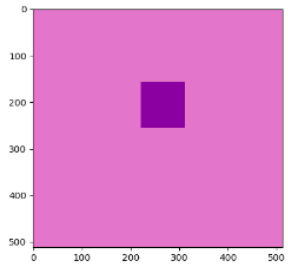
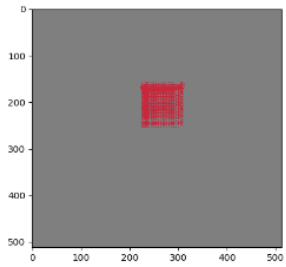
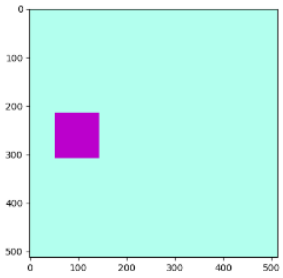
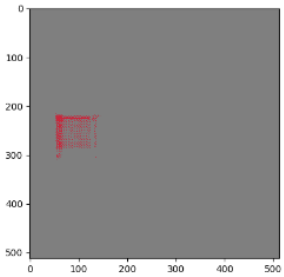
-Manoj Kumar Gootam





In the above picture, the top row contains the input images and the bottom row contains the corresponding output images. From the above examples, the image segmentation algorithm does a really good job.

The following are some images where simplified network does a bad job. The picture on the left side are the original images and the picture on the right side are the corresponding output images. There is significant loss in segmentation quality. One of the reasons could be that the Batch Normalization shifts the learned parameters and are not well tuned for these examples. So, some part of the segmentation is badly classified, however the boundary details are still clear.

6. CamVid Dataset

CamVid data : Cambridge-driving Labeled Video Database with 32 classes

367 Train and 233 Test Images

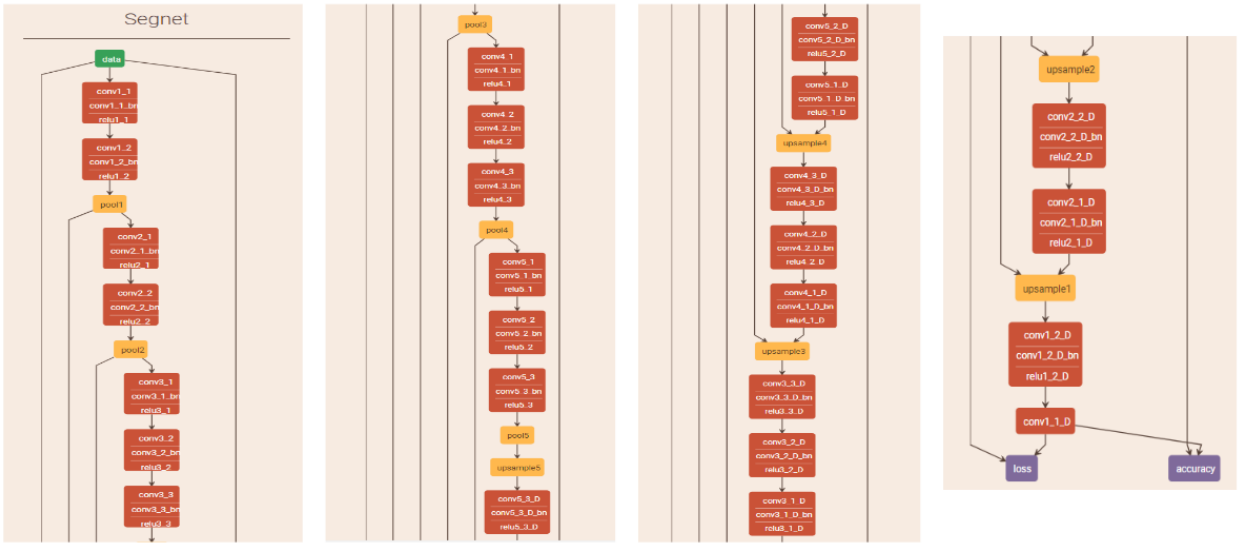
Each Image is of size 360x480

Class Labels : 11

Bicyclist, Sidewalk, Tree, Fence, Signs, Pedestrian, Car, Sky, Road, Building and Miscellaneous

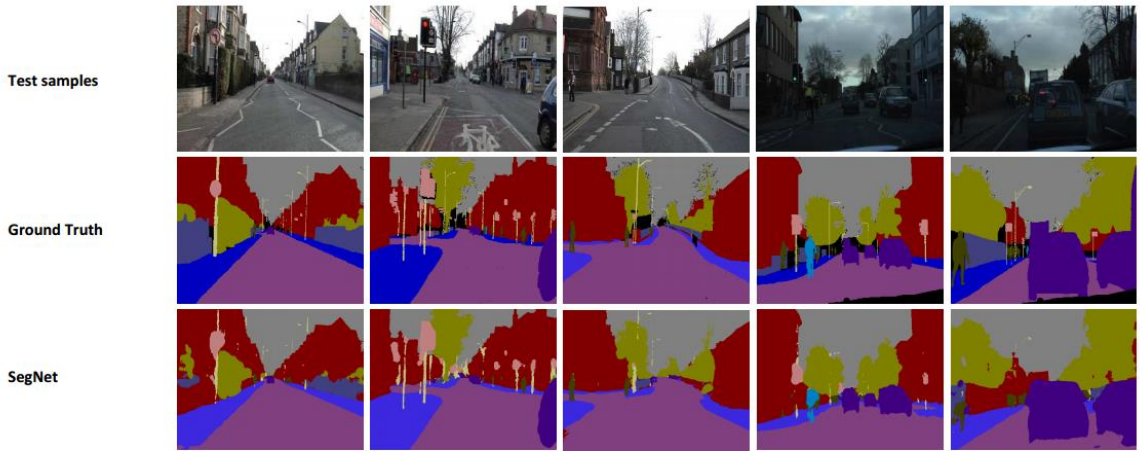
7. SegNet Architecture:

The following network architecture has 13 layers of encoders and 13 layers of decoders and a "SoftMaxWithLoss" layer towards the end that uses softmax and multinomial logistic loss.



Due to technological limitations, I could train the above algorithm for 100 iterations only and it does not converge. So, I tried using transfer learning by downloading the weights from the already implemented model and used it to train the network to produce the following outputs. In the following picture, the top row contains the actual images and the next row has the corresponding annotated images(Ground truth)

and the last row has the corresponding outputs from SegNet. For practical purposes, the following output seems quite perfect. The SegNet architecture claims to achieve 86% accuracy approximately.



8. Summary

The role of the decoder network is to map the low resolution encoder feature maps to full input resolution feature maps for pixel-wise classification. The novelty of SegNet lies in the manner in which the decoder upsamples its lower resolution input feature maps. Specifically, the decoder uses pooling indices computed in the max-pooling step of the corresponding encoder to perform non-linear upsampling. This eliminates the need for learning to upsample. The upsampled maps are sparse and are then convolved with trainable filters to produce dense feature maps. The following are some of the high level summary of the SegNet Architecture

- Many neural net that segment to pixel level have similar encoder -> decoder (deconvolution) steps, but SegNet have much more efficient (in speed and RAM) decoder step with little tradeoff in accuracy.

- SegNet's decoder uses data from encoder in a clever way. Other algorithm, decoder "learn" how to upsampling but it is slow and difficult to train. SegNet remembers just pointer data ("indices") that encoder produce as an input for decoder to upsampling, not the full output like some other networks.

- This means SegNet is more RAM friendly when use. It is upto 11 times more efficient than others.

- SegNet can be trained end-to-end. Some algorithm contains separate parts that you have to train individually, like the one that throw neural net output into separate CRF algorithm.

- SegNet removed fully connected final layers from VGG16 (very deep conv. net for image recognition) to reduce parameters, enable better end to end training. (VGG16 = 13 conv. + 3 fully connected layers, layer count down from 134M to 14.7M)

- "Competitive" performance on standard tests, but significantly faster.
- Max-pooling adds translation invariance, but too much decrease edge details.
- The author found that larger decoder is better. And using just indices from encoder as an input for decoder is a better way to upsampling.

9. Conclusion

This exercise is only a proof of concept for how simple convolutional networks can be used for identifying the segmentation of objects in an image. I would like to apply this to a business problem where image segmentation can be useful. For example, using this for a Home Décor Application, where the end user could look at the several designs for the interiors or home appliances, real time, by identifying the current objects in the image and replacing them with designs selected by the user. This is a good example for augmented reality and this approach is a viable solution for any other similar application, including self-driving cars.

10. References

http://cs231n.github.io/convolutional-networks/#fc - Stanford Course

http://mi.eng.cam.ac.uk/projects/segnet/tutorial.html - Caffe-Segnet

https://devblogs.nvidia.com/parallelforall/image-segmentation-using-digits-5/ - NVIDIA DIGITS 5

https://research.fb.com/learning-to-segment/ - Facebook

https://docs.google.com/presentation/d/10XodYojlW-1iurpUsMoAZknQMS36p7lVIfFZ-Z7V_aY/edit#slide=id.p - University of California Berkeley

http://cs.brown.edu/~pff/segment/ - Brown University

https://arxiv.org/pdf/1606.02147.pdf - Purdue University

11. Appendix

All my codes are uploaded on github along with the datasets. Please find the github link below:

https://github.com/manoj2312/deepLearning.

Solver Code: manoj_basic_solver.prototxt

Training Code: manoj_basic_train.prototxt

Inference Code: manoj_basic_inference.prototxt

Image Segemantation

-Manoj Kumar Gootam

Python Test Segmentation code: test_segmentation_camvid.py

If you plan to use “GPU”, please make the change from “CPU” to “GPU” in all the above files