Deep Convolutional Encoder-Decoder Architecture for Pixel-wise Image Segmentation

Yash Patel, Chandni Patel
Department of Computer Science
Illinois Institute of Technology
December 1, 2020

Abstract

The core trainable engine will be containing encoder network and corresponding decoder network which will consist of a hierarchy of decoders one corresponding to each encoder and then be followed by pixel-wise classification layer using Softmax. The architecture of the encoder network will have 13 convolutional layers. The architecture of decoder network will map the low-resolution encoder feature maps to original input resolution feature maps for pixel-wise classification using upsampling of lower resolution input feature maps. To perform non-linear upsampling in decoder, we will be using pooling indices from max-pooling step of the corresponding encoder for accurate boundary localization. The upsampling maps will be convolved with trainable filters to produce dense feature maps in decoder because upsampling maps are spared. It will be designed such that efficiency increases in terms of memory and computational time during inference when compared to competing architectures and also reduce number of trainable parameters without sacrificing performance.

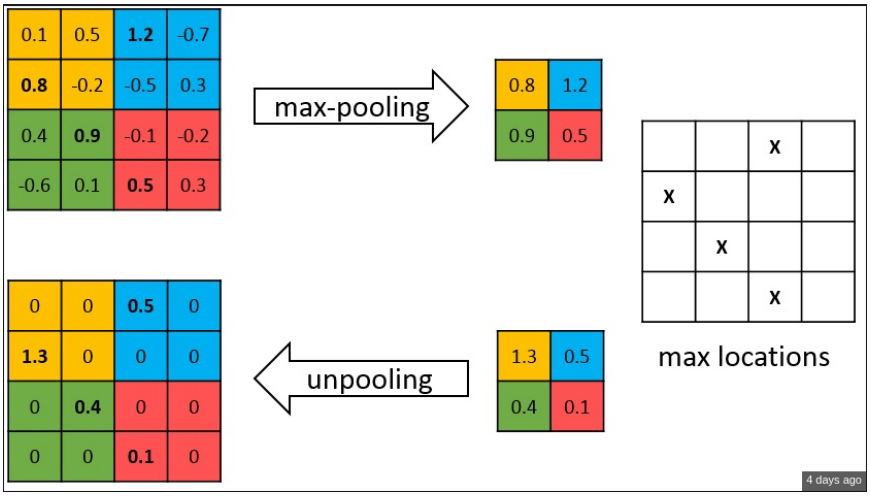
1 PROBLEM STATEMENT

Using semantic pixel-wise segmentation create deep fully convolutional neural network architecture. The main purpose will be to understand spatial relationship between classes like road, sidewalk from road scenes through modelling appearances (road, building) and shapes (car, pedestrians) and also retaining the boundary information from the extracted image representation.

2 PROPOSED SOLUTION

The solution is inspired by the unsupervised feature learning architecture proposed by Ranzato et al. [7]. The main learning segment is an encoder-decoder network. SegNet has an encoder network and a corresponding decoder network, followed by a final pixelwise classification layer. In SegNet, the encoder network is topologically identical to the convolutional layers in VGG16. Here, we removed the fully connected layers of VGG16. The advantages of that are: (i) the encoder network becomes significantly smaller, (ii) The network is easier to train than many other recent architectures. Discarding the fully connected layers also helps in retaining higher resolution feature maps at the deepest encoder output. This reduces the number of parameters in the SegNet encoder network significantly from 134 million to 14.7 million.

The main component of SegNet is the decoder network that consists of a hierarchy of decoders. Here, for each decoder we have corresponding encoder. To perform non-linear upsampling of the input feature maps, the decoders use the max-pooling indices received from the corresponding encoder. This idea was inspired from an architecture designed for unsupervised feature learning [7]. The advantages of reusing max-pooling indices in the decoding process are: (i) it improves boundary description, (ii) it reduces the number of parameters for end-to-end training, (iii) this upsampling method can be incorporated into any encoder-decoder architecture with a few modifications.



In the input images, we achieve translation invariance over small spatial shifts using max-pooling. In the feature map, sub-sampling results in a large input image context (spatial window) for each pixel. Several layers of max-pooling and sub-sampling can achieve more translation invariance for robust classification and correspondingly there is a loss of spatial resolution of the feature maps. In the encoder feature maps, it is necessary to capture and store boundary information before sub-sampling is performed [1]. If memory during inference is not constrained, then all the encoder feature maps can be stored after sub-sampling. A more efficient way to store this information involves storing only the max-pooling indices, i.e. in each pooling window, memorize the locations of the maximum feature value for each encoder feature map. The deep encoder-decoder network is trained jointly for a supervised learning task and hence the decoders are an integral part of the network in test time.

3 DATASET DETAILS

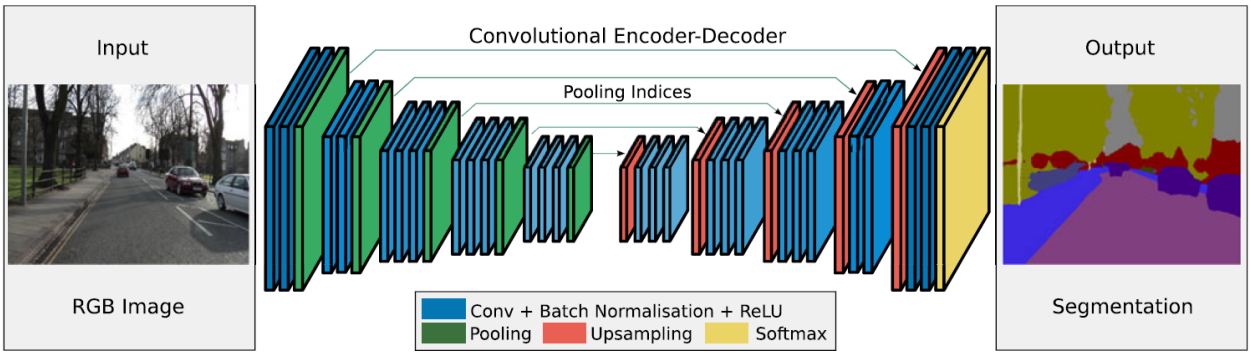
Cambridge-driving Labeled Video Database (CamVid) is used for training, validation, and testing. The database provides ground truth labels in order to associate each pixel with one of 32 semantic labels like road, car, pedestrian, sidewalk, trees, sky, building, etc. in image frames. The dataset consists of 369 training, 100 validation, and 232 testing RGB images at 960 x 720 resolution. Following are the classes and colors used to represent them:

Void	Building	Wall	Tree	VegetationMisc	Fence
Sidewalk	ParkingBlock	Column_Pole	TrafficCone	Bridge	SignSymbol
Misc_Text	TrafficLight	Sky	Tunnel	Archway	Road
RoadShoulder	LaneMkgsDriv	LaneMkgsNonDriv	Animal	Pedestrian	Child
CartLuggagePram	Bicyclist	MotorcycleScooter	Car	SUVPickupTruck	Truck_Bus
Train	OtherMoving				

Input images have 3 RGB channels. We resized the images to 128 x 128 resolution. We converted it to float32 datatype and normalized it between 0 to 1. For output images, we first convert pixel colors to class numbers and then apply one hot for K classes. Then randomly 50% images are augmented for trained. In augmentation, we apply horizontal flip, vertical flip, contrast, and brightness randomly to input images and flip corresponding output images.

4 MODEL ARCHITECTURE

SegNet has an encoder network and a corresponding decoder network, followed by a final pixelwise classification layer. This architecture is illustrated in Figure.



There are 13 convolutional layers in the encoder network which correspond to the first 13 convolutional layers in the VGG16 network [4] designed for object classification. Each encoder in the encoder network performs convolution with a filter bank to produce a set of feature maps. All of these layers are then batch normalized. After that, an element-wise rectified-linear non-linearity (ReLU) is applied. Then max-pooling with a 2 x 2 window and stride 2 is performed for non-overlapping window. The resulting output is sub-sampled by a factor of 2. For each sample, the indices of the max locations computed during pooling are stored and passed to the decoder.

The decoder network has 13 layers as each decoder layer is corresponding to each encoder layer. The final decoder output is taken as input for to a multi-class soft-max classifier to produce class probabilities for each pixel separately. The decoder up-samples the feature maps by using the stored pooled indices. To reconstruct the input image, it convolves this up-sampled map using a trainable decoder filter bank.

In the decoder network, using the memorized max-pooling indices from the corresponding encoder feature maps, each decoder up-samples its input feature maps. This step produces sparse feature maps which is SegNet decoding technique. These feature maps are then convolved with a trainable decoder filter bank to produce dense feature maps. Then a batch normalization step is applied to each of these maps. Unlike the other decoders in the network

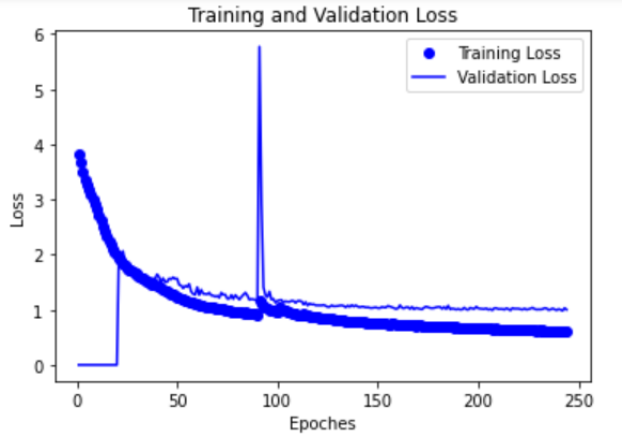
which produce feature maps with the same number of size and channels as their encoder inputs, the decoder corresponding to the first encoder closest to the input image produces a multi-channel feature map whereas its encoder input has three channels (RGB).

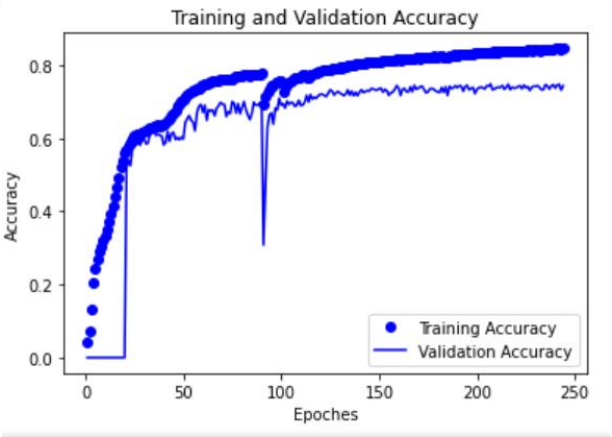
At the output of the final decoder, the high dimensional feature representation is fed to a trainable soft-max classifier. This soft-max classifies each pixel separately. The output of the soft-max classifier is a K channel image of probabilities where K is the number of classes [1]. The predicted segmentation corresponds to the class with maximum probability at each pixel. Finally, we have 29 million parameters for SegNet model,

For training, the encoder and decoder weights were all initialized using he uniform weight initializer. we used the SGD optimizer with a fixed learning rate of 0.001 and momentum of 0.9 and decay 0. Stochastic gradient descent (SGD) gives the ability to train end-to-end in order to jointly optimize all the weights in the network using an efficient weight update technique which is an additional benefit as it is more easily repeatable. We ran 244 epochs to train our model due to time, memory, and processing constraints as this took more than 3 days. We use the categorical cross-entropy loss as the objective function for training the network. In a mini-batch, the loss is summed up over all the pixels.

There is a need to weight the loss differently based on the true class, when there is large variation in the number of pixels in each class in the training set. For example, road, sky and building pixels dominate the CamVid dataset. It is called class balancing. We use median frequency balancing where the weight assigned to each class in the loss function is the ratio of the median of class frequencies computed on the entire training set divided by the class frequency. In the training set larger classes have a weight smaller than 1 and the weights of the smallest classes are the highest.

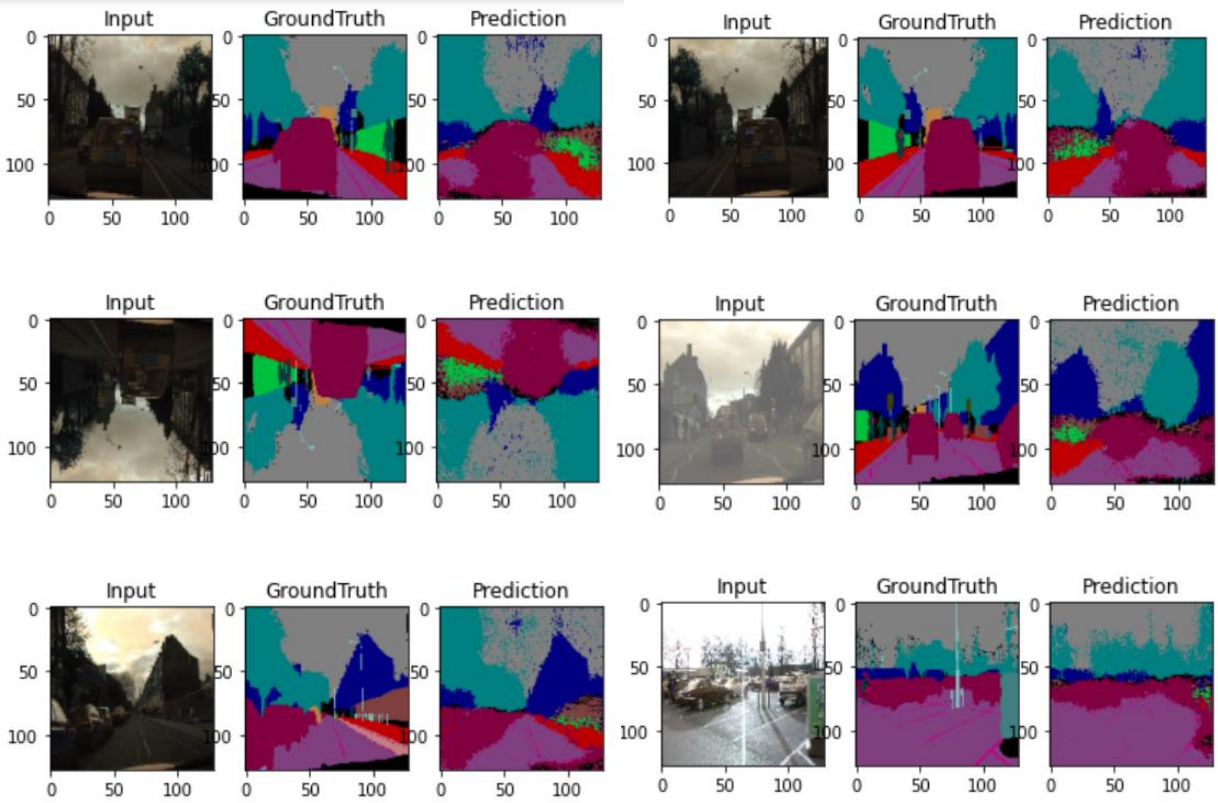
Here, we have plotted the loss and accuracy for epochs by saving the history at every run. Refer the following graphs:

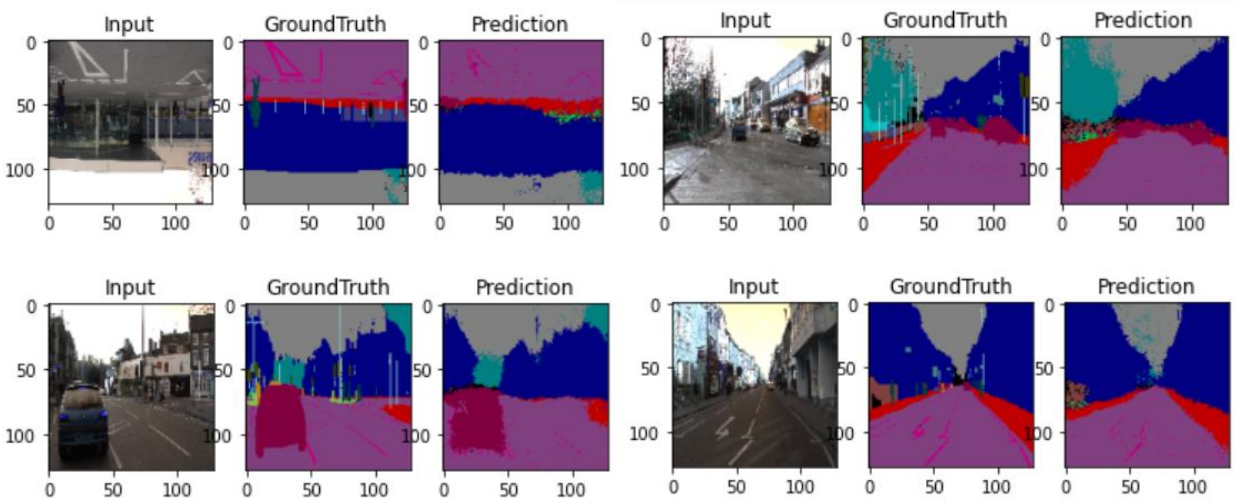




5 RESULT AND EVALUATION

Here are the test results that we obtained on the model we trained along with the input image and ground truth for comparison:





Here are the precision, recall, F1-score, and support for all the classes:

	precision	recall	fscore	support
Animal	0.000000	0.000000	0.000000	292.0
Archway	0.000000	0.000000	0.000000	2858.0
Bicyclist	0.000000	0.000000	0.000000	28753.0
Bridge	0.000000	0.000000	0.000000	2622.0
Building	0.831025	0.958651	0.890287	1386398.0
Car	0.481220	0.759798	0.589242	188350.0
CartLuggagePram	0.000000	0.000000	0.000000	1575.0
Child	0.000000	0.000000	0.000000	1933.0
Column_Pole	0.000000	0.000000	0.000000	62076.0
Fence	0.425258	0.575621	0.489145	100891.0
LaneMkgsDriv	0.494901	0.224109	0.308512	111727.0
LaneMkgsNonDriv	0.000000	0.000000	0.000000	687.0
Misc_Text	0.000000	0.000000	0.000000	39921.0
MotorcycleScooter	0.000000	0.000000	0.000000	336.0
OtherMoving	0.000000	0.000000	0.000000	28324.0
ParkingBlock	0.000000	0.000000	0.000000	21259.0
Pedestrian	0.142857	0.000213	0.000426	37505.0
Road	0.916838	0.942197	0.929344	1703076.0
RoadShoulder	0.000000	0.000000	0.000000	14801.0
Sidewalk	0.774300	0.837143	0.804496	413240.0
SignSymbol	0.000000	0.000000	0.000000	7765.0
Sky	0.970022	0.810416	0.883065	928149.0
SUVPickupTruck	0.000000	0.000000	0.000000	32185.0
TrafficCone	0.000000	0.000000	0.000000	249.0
TrafficLight	0.000000	0.000000	0.000000	22038.0
Train	0.000000	0.000000	0.000000	0.0
Tree	0.693602	0.940159	0.798276	615077.0
Truck_Bus	0.000000	0.000000	0.000000	11230.0
Tunnel	0.000000	0.000000	0.000000	1.0
VegetationMisc	0.209281	0.009059	0.017366	49787.0
Void	0.350318	0.185987	0.242976	158140.0
Wall	0.421411	0.393641	0.407053	74451.0

```
evaluation of segnet on training data :
369/369 [==============================] - 195s 529ms/step
loss : 0.7173007361611053
accuracy : 0.8097500801086426
```

```
evaluation of segnet on validation data :
100/100 [==============================] - 52s 525ms/step
loss : 1.0021786427497863
accuracy : 0.7431433200836182
```

```
evaluation of segnet on testing data :
232/232 [==============================] - 127s 547ms/step
loss : 1.1325214772388852
accuracy : 0.6955529451370239
```

The accuracy can still improve with more epochs, which can be implemented further. Due to uneven class frequency, we used class weights and median frequency balancing during training because there is large variation in the number of pixels in each class in the training set and to weight the loss differently based on the true class.

6 CONCLUSION

SegNet is efficient for both memory and computational time. When comparing to other models, this model has smaller size (112 MB), fewer parameters (29,461,088), and increased training speed. If we got more time to train our model, then accuracy can be increased to at least 90% with more epochs. In SegNet the practical trade-offs involved designing architectures for segmentation, particularly training time, memory versus accuracy. Each epoch takes longer time when compared to other model, but overall training time is faster to get required accuracy. The architectures which store the encoder network feature maps in full usually perform best, but they consume more memory during inference time. SegNet is more efficient as it only stores the max-pooling indices of the feature maps and uses them in decoder network to achieve good performance. On large and well-known datasets, SegNet performs competitively, achieving high scores for road scene understanding. Here, we lacked time, processing power, and had memory issues to get required accuracy.

7 TEAM MEMBER RESPONSIBILITIES

Both team members work on understanding the research paper and SegNet model, wrote final report and prepared presentation. Yash worked on model creation and wrote all code for that. Chandni worked on training, evaluation, and analysis part. Testing was done by both.

8 REFERENCES

[1] 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, doi: 10.1109/TPAMI.2016.2644615

[2] G. Brostow, J. Fauqueur, and R. Cipolla, “Semantic object classes in video: A high-definition ground truth database,” *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, 2009.

[3] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” *in Proc. IEEE Int. Conf. Comput. Vis*., 2015, pp. 1520–1528.

[4] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv:1409.1556*, 2014.

[5] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[6] V. Badrinarayanan, A. Handa, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling,” *arXiv preprint arXiv:1505.07293*, 2015.

[7] M. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun, “Unsupervised learning of invariant feature hierarchies with applications to object recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2007, pp. 1–8.

[8] Link for the dataset: https://drive.google.com/drive/folders/1rE23coR6ddOWOPtg4oB6qeIVzmYzvGt4?usp=sharing