Semantic Segmentation on Cityscape Dataset

Anshuman Gaharwar, Shivani Junawane, Sanjay Shanbhag, Ashish Jain

Abstract

Semantic Segmentation deals with assigning classes to every pixel in the image to get a pixel-wise dense classification. In this project, we try to solve the problem of semantic segmentation using the Cityscapes dataset, where we segment 10 classes/objects in the given image. Our main aim is to be able to segment objects like vehicles and pedestrians in street images, as it is one of the major components of autonomous driving. People have been working on semantic segmentation/scene parsing problems since 2007, but the major breakthroughs came after the introduction of fully convolutional networks (CNN) in early 2014. These fully convolutional networks were able to achieve very high accuracies on benchmark datasets like PASCAL VOC and MSCOCO. As part of our course project, we trained our own implementation of the UNet model and achieved an IOU score of 0.5242 on the train set and 0.4536 on the test set.

I. INTRODUCTION

A. Background

Since the introduction of fully convolutional networks for semantic segmentation and scene understanding tasks, many different network architectures have been developed and have proven to work well with common datasets like PASCAL VOC and MSCOCO. There are many applications of scene understanding like autonomous navigation, assisting the partially sighted, medical diagnostic, image editing, etc. Before the deep learning era techniques such as TextonBoost, TextonForest, Conditional Random Forest-based approaches were used for solving the semantic segmentation problem.

B. Related Work

In [1] authors propose a Conditional Random Field Model of Object Classes since use of a Conditional Random Field allows us to incorporate shape, texture, color, location and edge cues in a single unified model. The authors try to learn the model parameters by maximizing the conditional likelihood of the true class labels given the training data. This can be achieved using gradient ascent, and computing the gradient of the likelihood with respect to each parameter, requiring the evaluation of marginals over the class labels for each training image.

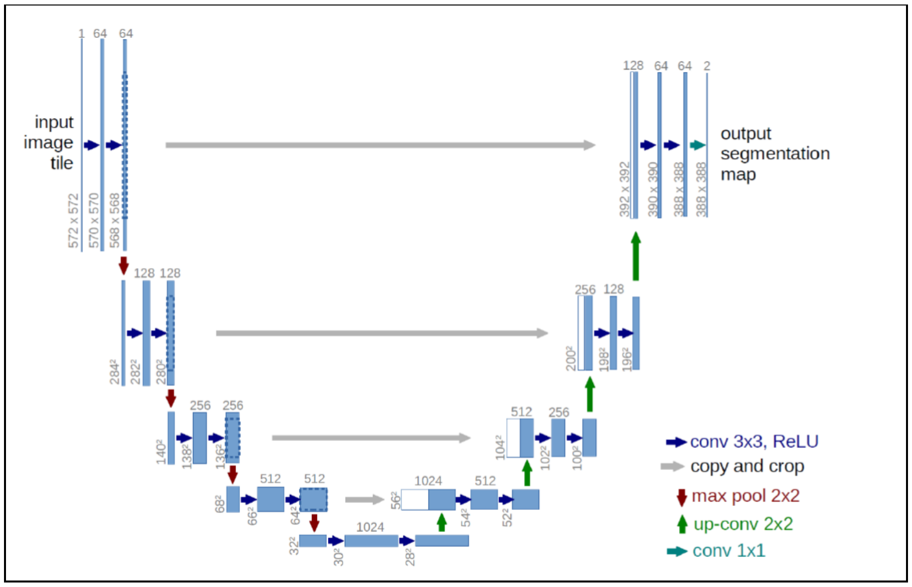
In [2] authors presented ensembles of decision trees that act directly on image pixels, and therefore do not need the expensive computation of filter-bank responses or local descriptors. the bag of words model in which a histogram of visual words is created over the whole image or a region of interest, by discarding spatial layout. The histogram is used as input to a classifier to recognize object categories. For segmentation, image-level prior emphasizes those categories that the automatic categorization believes to be present.

In [3] authors present superparsing which works by scene-level matching with global image descriptors, followed by superpixel-level matching with local features and efficient Markov random field (MRF) optimization and this method was found to be faster & better than SIFT since there was no training required. Given a query image, we retrieve similar images from our dataset using several global features. Next, we divide the query into superpixels and compute a per-superpixel likelihood ratio score for each class based on nearest neighbor superpixel matches from the retrieval set. These scores, in combination with a contextual MRF model, give a dense labeling of the query image.

In [4] authors propose a U net architecture that works with very few images and yields more precise segmentations. By using an overlap-tile method, this strategy enables for the smooth segmentation of arbitrarily large images. The missing context is extrapolated by mirroring the input image to forecast the pixels in the image's border region. It uses a contracting and expansive process that allows for seamless segmentation of large images by an overlap tile strategy.

II. METHOD

A. Network



Source: Olaf Ronneberger , Philipp Fischer, Thomas Brox "U Net: Convolutional Networks for Biomedical Image Segmentation", MICCAI, 2015

Figure 1 illustrates the network design. It is made up of a contracting path (on the left) and an expansive path (on the right) (right side). The convolutional network's contracting path follows the standard architecture. It consists of two 3x3 convolutions (unpadded convolutions) that are applied repeatedly, each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling[4]. We double the number of feature channels with each downsampling step[4]. An upsampling of the feature map is followed by a 2x2 convolution (up-convolution) that halves the number of feature channels, a concatenation with the proportionally cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU[4]. Due to the loss of boundary pixels in every convolution, cropping is required[4]. A 1x1 convolution is utilized at the nal layer to map each 64-component feature vector to the desired number of classes. The network comprises a total of 23 convolutional layers[4].

B. Loss Functions

CrossEntropyLoss:
It is used to determine the precision of a machine learning or deep learning model by quantifying the difference between the estimated probability and the intended outcome. Essentially, this type of loss function assesses the performance of your model by converting its variables to real values and calculating the "loss" associated with them. The greater the gap between the two, the greater the loss. We define CrossEntropyLoss as below where y is the ground truth vector and ŷ (or some other value taken directly from the last layer output) is the estimate.

$$L(y,\hat{y}) = -\Sigma\, y.\, \log(\hat{y})$$

C. K Nearest Neighbours:

The KNN algorithm predicts the values of new data points based on 'feature similarity,' which implies that the new data point will be assigned a value depending on how closely it matches the points in the training set. The KNN algorithm is a sort of supervised machine learning method that may be used to solve both classification and regression predicting problems. However, in industry, it is mostly utilized to solve classification and prediction problems.

We have used this algorithm to combine similar predictions to one prediction (as our different intensity values are different labels). Ex: combine all predictions from 6.5 to 7.5 to the label 7

D. Evaluation metric:

Intersection over Union: Intersection over Union is used for determining how accurate an item detector is on a given dataset. IoU can be used to evaluate any method that produces anticipated bounding boxes as an output. We'll need the ground-truth bounding boxes and our model's anticipated bounding boxes to use IoU. We define IoU as follows:

$$\text{IoU} = \frac{\textit{Area of Overlap}}{\textit{Area of Union}}$$

III. RESULTS

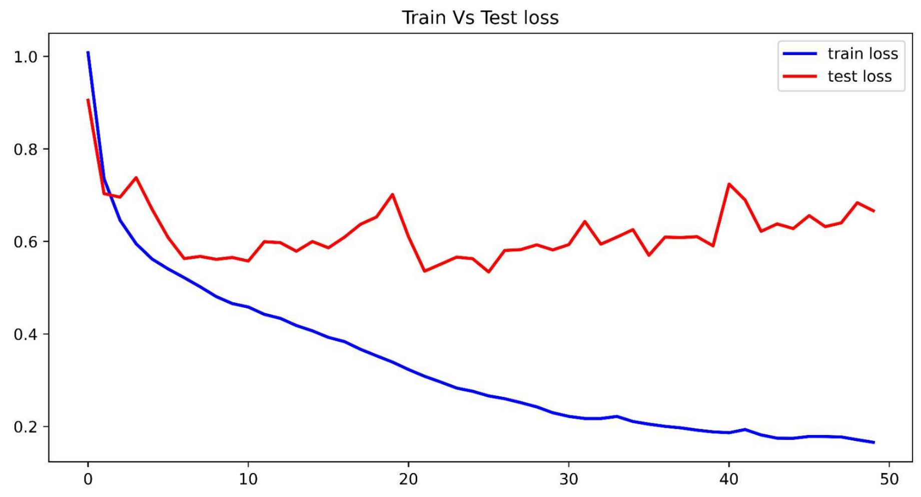
We trained our UNet model with CrossEntropyLoss and Adam optimizer with learning rate as 0.001 for 50 epochs with the train set containing 2975 images and evaluated our model on the test set containing 500 images.
Dataset: https://www.kaggle.com/dansbecker/cityscapes-image-pairs

Best/Final Model: No of Training Epochs: 22
Train set: Loss: 0.3086
IoU: 0.5242
Test set: Loss: 0.5357
IoU: 0.4536

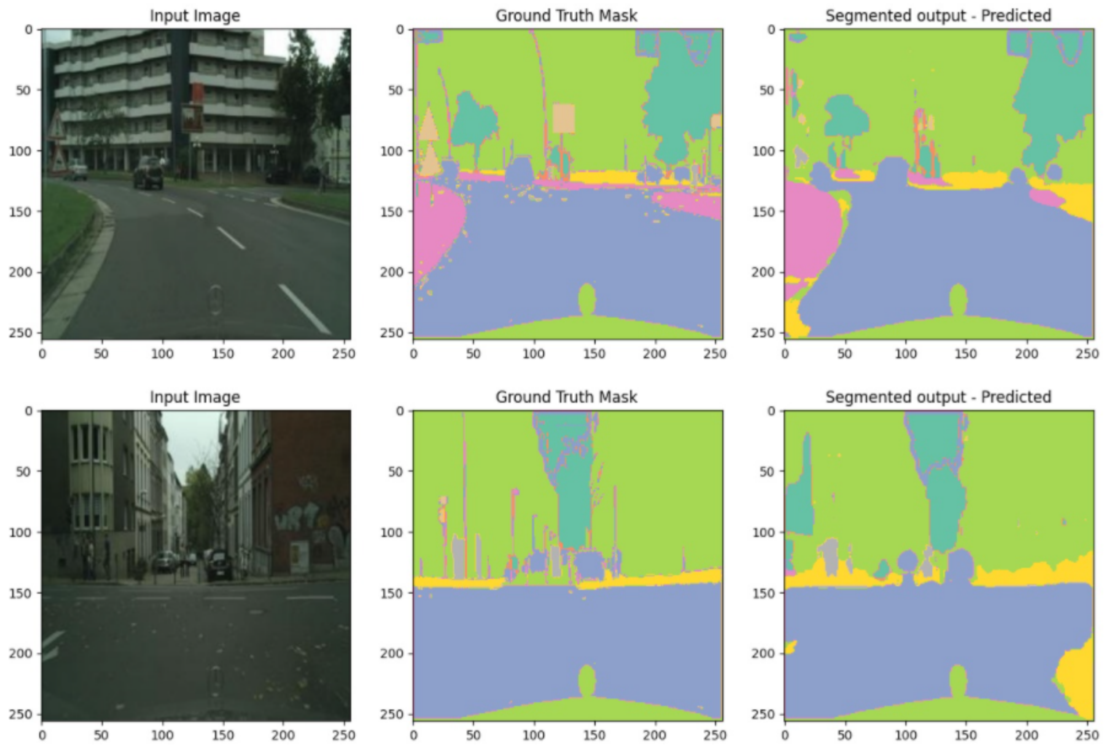
Usage:

```
python3 train_evaluate.py --datasetHome ./cityscapes_data --batch_size 16
--num_epochs 50 --learning_rate 0.001
```

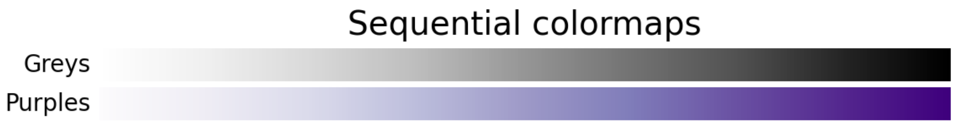
Below is the graph of train and test loss over 50 epochs.



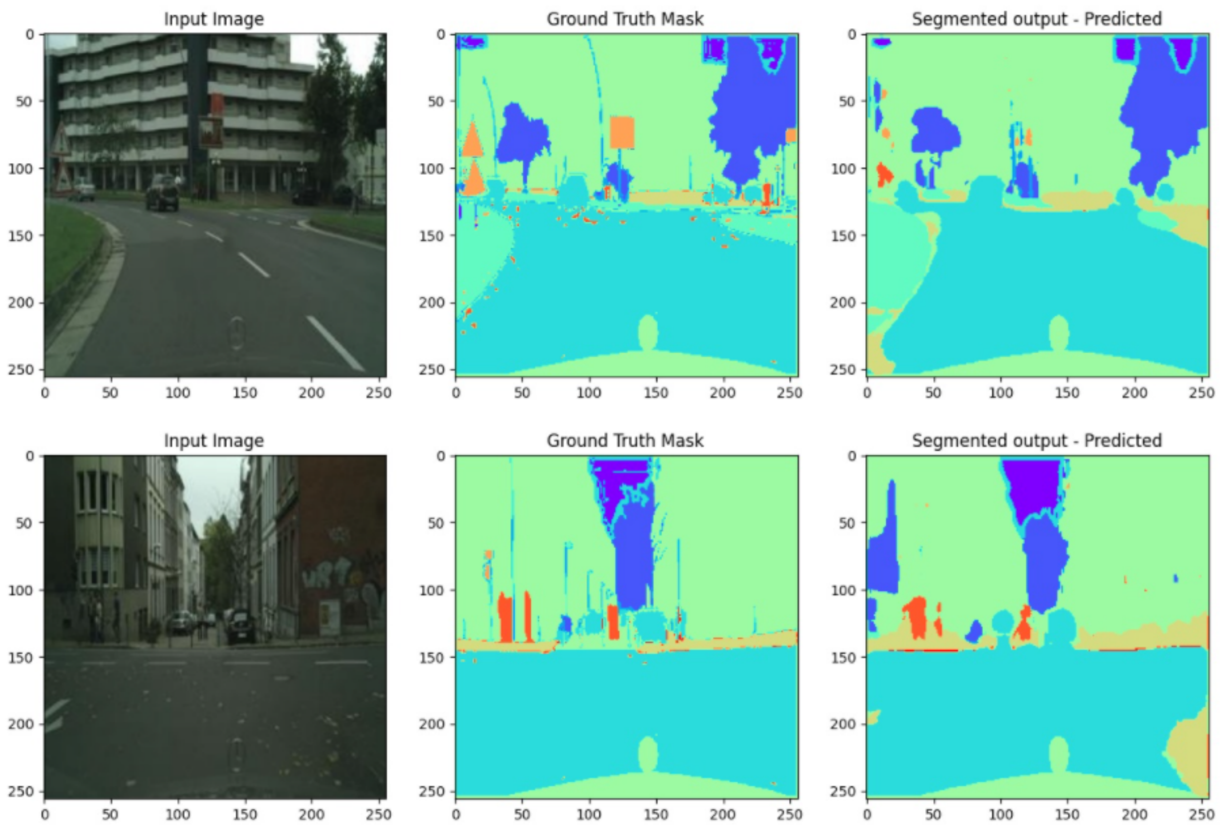
Segmentation Results:



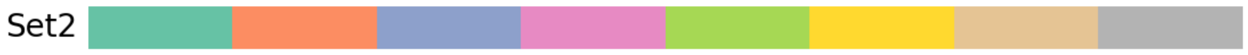
In the above figure, we have plotted the results of the prediction from our segmentation model using the "Set2" cmap of matplotlib for visualization. There exist many options to visualize the results but most of the cmap configurations (Greys, Purple etc) change the color intensity gradually as shown in the next figure.

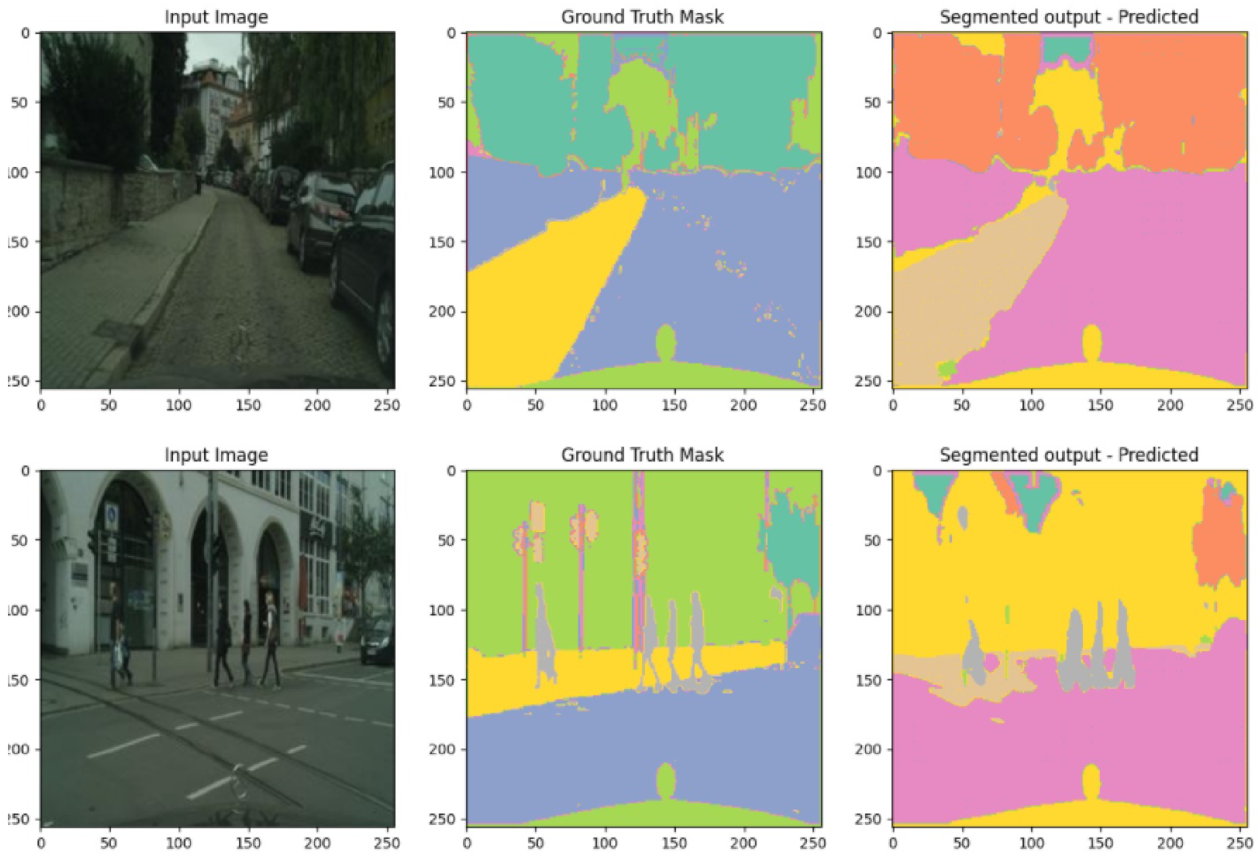


Output from sequential + diverging color map:



As our consecutive predicted values were from different classes (i.e road label is 7 and sidewalk label is 8, even if our model predicts right values, cmap will plot those labels with very close intensity), we were not getting clean visualization (Observe small difference between sidewalk and road labels). Thus, we selected a qualitative colormap for visualization, which provides different intervals to be shown in different contrasting colors.





In the above figure, there were some labels which got plotted in neighbouring colors in the visualization. Although, the prediction from our model was correct (evident from IoU score).

IV. DISCUSSION AND ANALYSIS

Results could be made better by:

- Adding more data:
  Originally, we had selected data for CVPR autonomous driving with size of approx 100GB, but, due to resource and time crunch we later switched to a smaller dataset(Cityscapes). We can get better results by training on the former dataset.

- Training for more epochs:
  We can expect better outcomes by training for more epochs considering we have a bigger dataset. We did not train for higher epochs in order to avoid overfitting.

- Use of multiple GPU's for training:
  Higher batch sizes can be trained utilizing several GPUs and data parallel training. We can clone the same model instance to multiple GPUs and split the batch data between them. Each GPU will create an output for the data it was given, and the ultimate loss can be estimated by aggregating these distributed losses. This will help us to get quicker results

V. CONCLUSION AND FUTURE WORK

In this work we implemented the UNet model to segment objects like vehicles and pedestrians in street images, as it is one of the major components of autonomous driving. We observed that our model has a comparatively low loss value after training for 50 epochs and in the test set we observed that at 22nd epoch the testing loss was lowest. Semantic segmentation is a very useful task in fields of autonomous driving, Geo-sensing, facial segmentation and medical image processing.

VI. CONTRIBUTIONS

1. Anshuman Gaharwar:

- Dataset pre-processing
- Trainer scripts
- Local GPU+CUDA setup after Newton space was exhausted
- Hook implementations for continuous monitoring of model
- Generalized model builder
- Predicted output to plots converter

2. Shivani Junawane

- First dataset finalizing
- Segnet Architecture study
- Model scripts
- Second dataset finalizing
- Model training and monitoring on Newton
- Evaluation script

3. Sanjay Shanbhag

- First Dataset curation
- First dataset visualization
- Model training and monitoring on local setup
- Final code integration
- Fine tuning models for best IoU and min loss

4. Ashish Jain

- Problem Statement Proposal
- U-Net architecture study
- Reference papers literature survey
- Second dataset curation + visualization
- Report integration

REFERENCES:

1. J. Shotton, J. Winn, C. Rother, and A. Criminisi, TextonBoost: Joint Appearance, Shape And Context Modeling For Multi-class Object Recognition And Segmentation, ECCV 2006.
2. “Semantic Texton Forests for Image Categorization and Segmentation”, 2008
3. SuperParsing (J. Tighe and S. Lazebnik, SuperParsing: Scalable Nonparametric Image Parsing with Superpixels, ECCV 2010)
4. Olaf Ronneberger, Philipp Fischer, Thomas Brox “U Net: Convolutional Networks for Biomedical Image Segmentation”, MICCAI, 2015
5. Youtube Video: CAP5415 Lecture 25 [Semantic Segmentation] - Fall 2021 (https://www.youtube.com/watch?v=te_0SBbS3uA)
6. Youtube Video: https://www.youtube.com/watch?v=H1SGhL4flIQ