

Machine Learning Engineer Nanodegree

Capstone Proposal

Sankirna Joshi

September 3rd, 2017

Identify Car Boundary and Remove its background from an image

Domain Background

In computer vision, the process of partitioning a digital image into multiple segments (sets of pixels, also known as super-pixels) is called as [image segmentation](#) or semantic segmentation. Image segmentation has been used in various fields such as object detection, face recognition and medical imaging. In this [Kaggle competition](#), we will use image segmentation techniques to attempt to automatically detect the car boundary and remove any background from the image.

There are various background removal tools that are available online such as [this](#), but they often rely on human input to select regions to keep and regions to discard. It would be great if we could using machine learning, completely automate this task. This would make it very easy for Carvana to apply new backgrounds to their car images and thereby remove their dependency on photo editors which need skill persons to operate and are time consuming.

Before CNNs were popular, image segmentation was performed by approaches like [TextonForest](#) and [Random Forests](#). Initial Deep learning approaches included [patch classification](#) where each pixel was separately classified using the surrounding patch. As CNNs gained popularity, various CNN architectures developed for image segmentation such as [FCN](#), [SegNet](#), [DensNet](#) and are currently the state-of-the-art techniques in semantic image segmentation.

Problem Statement

Background removal is currently a manual task or at best, a semi-automatic technique. We want to develop a deep learning algorithm that automatically removes the background from the car images by use of deep learning techniques.

Datasets and Inputs

The dataset to this problem is acquired from [here](#). This data is hosted on Kaggle as part of 'Carvana Image Masking Challenge' competition. We have been provided with high resolution images as well as lower resolution images. In the training data, we have 318 unique cars and each car has exactly 16 images, each one taken at different angles. So a total of 5088 training car images. For each of these 5088 images we have a supplementary .gif file that contains the manually cutout mask for each image. We also have a csv file containing the run-length encoded version of the training set masks and a file containing metadata info such as care make, year as well. Our model will be trained and validated on this data and tested on the test data provided separately on the above link.

Solution Statement

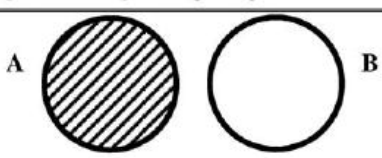
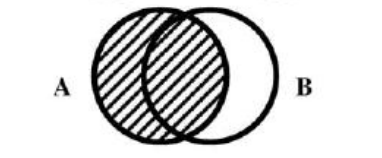

I plan to use deep learning techniques as a solution to this problem. I will approach the problem as a classification task as we need to predict which pixels which will lie in the mask cutout of the car and which will lie outside it. That is, we need to classify every pixel in one of the two classes. I plan to use a CNN based model to propose a solution for this problem.

Benchmark Model

As a benchmark model, I plan to train on one of the contemporary image classification architectures like VGG-16 and compare the results with my model. We will need to tweak the basic image classification algorithm a little as we need to not only identify the object (car in our case) but also identify its location in the image. Max pooling layers would ideally need to be removed as they reduce resolution and make image coarser.

Evaluation Metrics

Dice Coefficient will be used to evaluate the test results. The Dice coefficient can be used to compare the pixel-wise agreement between a predicted segmentation and

Spatial Overlap of Target Segmetations A and B	$DSC=2(A \cap B)/(A+B)$
	No Overlap: $DSC=0$
	Partial Overlap: $0 < DSC < 1$
	Complete Overlap: $DSC=1$

corresponding ground truth.

The formula is given by:

$$Dice(A, B) = \frac{2|A \cdot B|}{|A| + |B|}$$

[Image source](#)

Project Design

- **Programming Requirements**

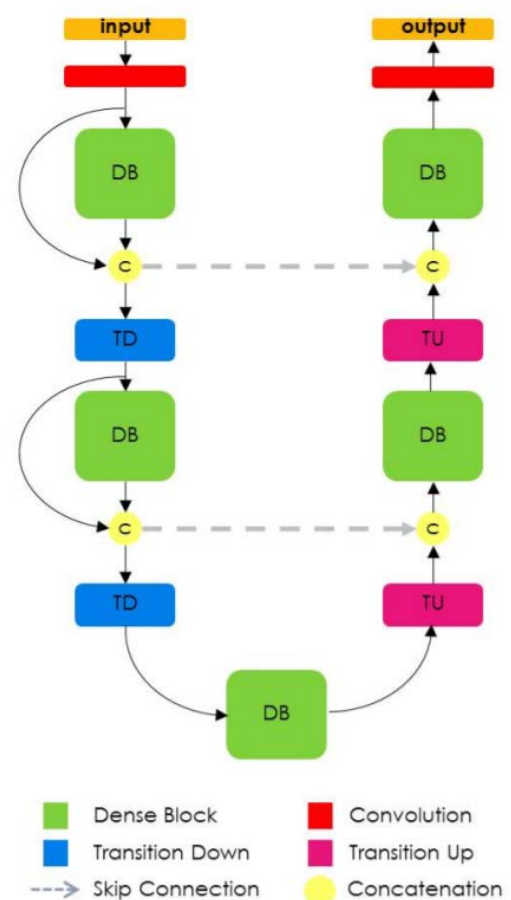
- Python 3.5 and various essential libraries including
 - scikit-learn – Open source ML library for Python.
 - Keras – Open source NN library for Python.
 - Tensorflow – Open Source DL library for Python.
- Jupyter Notebooks – Open Source web app/ editor to run code.

- **Data**

- Training and Testing data has been made available on Kaggle [here](#).
- Training Data will be split into 80:20 split for performing training and validation respectively.
- Kaggle provides testing data separately. Testing will be performed on this data.

- **Model**

- As discussed earlier, I will be implementing a deep learning model that has enhances over the image classification architecture.
- Our problem has a constant background (studio) and changing foreground objects (cars) as seen in the example image below. [Tiramisu](#) model seems to be an appropriate candidate to try as this model performs well on multiclass image segmentation on similar [CamVid](#) dataset which has a more or less constant background (roads/streets) and changing foreground objects.
- The Tiramisu model architecture is shown in the adjacent image.
- It consists of Dense blocks which is a concatenation of previous feature maps and are placed on both down-sampling and up-



sampling paths. In the down-sampling path, the input to Dense block is concatenated with the output whereas in up-sampling it is not.

- Skip connections are used to compensate the resolution loss due to pooling.
- The model will be trained on parameters similar to that that are defined in the paper and findings will be reported.
 - Loss function: Cross Entropy loss
 - Optimizer: RMSprop
 - Learning Rate: 0.001

Example training image from the Carvana dataset.

