```
%matplotlib inline
import cv2
from pylab import *

rcParams['figure.figsize'] = 15, 15

import torch
from torch import nn
from unet_models import unet11
from pathlib import Path
from torch.nn import functional as F
from torchvision.transforms import ToTensor, Normalize, Compose

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

def get_model():
    model = unet11(pretrained='carvana')
    model.eval()
    return model.to(device)

def mask_overlay(image, mask, color=(0, 255, 0)):
    """
    Helper function to visualize mask on the top of the car
    """
    mask = np.dstack((mask, mask, mask)) * np.array(color)
    mask = mask.astype(np.uint8)
    weighted_sum = cv2.addWeighted(mask, 0.5, image, 0.5, 0.)
    img = image.copy()
    ind = mask[:, :, 1] > 0
    img[ind] = weighted_sum[ind]
    return img

def load_image(path, pad=True):
    """
    Load image from a given path and pad it on the sides, so that eash side is
    divisible by 32 (newtwork requirement)

    if pad = True:
        returns image as numpy.array, tuple with padding in pixels as(x_min_pad,
        y_min_pad, x_max_pad, y_max_pad)
    else:
        returns image as numpy.array
    """
    img = cv2.imread(str(path))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    if not pad:
        return img

    height, width, _ = img.shape

    if height % 32 == 0:
        y_min_pad = 0
        y_max_pad = 0
    else:
        y_pad = 32 - height % 32
        y_min_pad = int(y_pad / 2)
        y_max_pad = y_pad - y_min_pad

    if width % 32 == 0:
        x_min_pad = 0
        x_max_pad = 0
    else:
        x_pad = 32 - width % 32
        x_min_pad = int(x_pad / 2)
        x_max_pad = x_pad - x_min_pad
```

```
65
66      img = cv2.copyMakeBorder(img, y_min_pad, y_max_pad, x_min_pad, x_max_pad,
        cv2.BORDER_REFLECT_101)
67
68      return img, (x_min_pad, y_min_pad, x_max_pad, y_max_pad)
69
70  img_transform = Compose([
71      ToTensor(),
72      Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
73  ])
74
75  def crop_image(img, pads):
76      """
77      img: numpy array of the shape (height, width)
78      pads: (x_min_pad, y_min_pad, x_max_pad, y_max_pad)
79
80      @return padded image
81      """
82      (x_min_pad, y_min_pad, x_max_pad, y_max_pad) = pads
83      height, width = img.shape[:2]
84
85      return img[y_min_pad:height - y_max_pad, x_min_pad:width - x_max_pad]
86
87  model = get_model()
88
89  # Example on Carvana dataset car
90  img, pads = load_image('lexus.jpg', pad=True)
91  imshow(img)
92  with torch.no_grad():
93      input_img = torch.unsqueeze(img_transform(img).to(device), dim=0)
94  with torch.no_grad():
95      mask = F.sigmoid(model(input_img))
96  mask_array = mask.data[0].cpu().numpy()[0]
97  mask_array = crop_image(mask_array, pads)
98  imshow(mask_array)
99  imshow(mask_overlay(crop_image(img, pads), (mask_array > 0.5).astype(np.uint8)))
```