car model

May 9, 2018

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from skimage.io import imread
        from skimage.transform import downscale_local_mean
        from os.path import join
        from tqdm import tqdm_notebook
        import cv2
        from sklearn.model_selection import train_test_split
```

```
In [5]: input_folder = join('input')

        df_mask = pd.read_csv(join(input_folder, 'train_masks.csv'), usecols=['img'])
        ids_train = df_mask['img'].map(lambda s: s.split('_')[0]).unique()

        imgs_idx = list(range(1, 17))
```

```
In [6]: load_img = lambda im, idx: imread(join(input_folder, 'traincar', '{}_{:02d}.jpg'.format(
        load_mask = lambda im, idx: imread(join(input_folder, 'train_masks', '{}_{:02d}_mask.gif
        resize = lambda im: downscale_local_mean(im, (4,4) if im.ndim==2 else (4,4,1))
        mask_image = lambda im, mask: (im * np.expand_dims(mask, 2))
```

```
In [7]: num_train = 32  # len(ids_train)

        # Load data for position id=1
        X = np.empty((num_train, 320, 480, 12), dtype=np.float32)
        y = np.empty((num_train, 320, 480, 1), dtype=np.float32)

        with tqdm_notebook(total=num_train) as bar:
            idx = 1 # Rotation index
            for i, img_id in enumerate(ids_train[:num_train]):
                imgs_id = [resize(load_img(img_id, j)) for j in imgs_idx]
                # Input is image + mean image per channel + std image per channel
                X[i, ..., :9] = np.concatenate([imgs_id[idx-1], np.mean(imgs_id, axis=0), np.std
                y[i] = resize(np.expand_dims(load_mask(img_id, idx), 2)) / 255.
                del imgs_id # Free memory
                bar.update()
```

```
HBox(children=(IntProgress(value=0, max=32), HTML(value='')))
```

```
In [8]: X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
In [9]: y_train_mean = y_train.mean(axis=0)
        y_train_std = y_train.std(axis=0)
        y_train_min = y_train.min(axis=0)

        y_features = np.concatenate([y_train_mean, y_train_std, y_train_min], axis=2)

        X_train[:, ..., -3:] = y_features
        X_val[:, ..., -3:] = y_features
```

```
In [10]: X_mean = X_train.mean(axis=(0,1,2), keepdims=True)
         X_std = X_train.std(axis=(0,1,2), keepdims=True)

         X_train -= X_mean
         X_train /= X_std

         X_val -= X_mean
         X_val /= X_std
```

```
In [11]: from keras.layers import Conv2D
         from keras.models import Sequential
         import keras.backend as K

         model = Sequential()
         model.add( Conv2D(16, 3, activation='relu', padding='same', input_shape=(320, 480, 12)
         model.add( Conv2D(32, 3, activation='relu', padding='same') )
         model.add( Conv2D(1, 5, activation='sigmoid', padding='same') )
```

```
/home/nikhil/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversio
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

```
In [12]: from keras.optimizers import Adam
         from keras.losses import binary_crossentropy

         smooth = 1.

         # From here: https://github.com/jocicmarko/ultrasound-nerve-segmentation/blob/master/tr
         def dice_coef(y_true, y_pred):
             y_true_f = K.flatten(y_true)
             y_pred_f = K.flatten(y_pred)
```

```
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)


def bce_dice_loss(y_true, y_pred):
    return 0.5 * binary_crossentropy(y_true, y_pred) - dice_coef(y_true, y_pred)

model.compile(Adam(lr=1e-3), bce_dice_loss, metrics=['accuracy', dice_coef])
```

```
In [14]: history = model.fit(X_train, y_train, epochs=15, validation_data=(X_val, y_val), batch_
```

```
Train on 25 samples, validate on 7 samples
Epoch 1/15
 - 18s - loss: -5.1759e-02 - acc: 0.7565 - dice_coef: 0.3500 - val_loss: -2.2646e-01 - val_acc:
Epoch 2/15
 - 18s - loss: -3.4551e-01 - acc: 0.8516 - dice_coef: 0.5267 - val_loss: -4.6521e-01 - val_acc:
Epoch 3/15
 - 18s - loss: -5.7210e-01 - acc: 0.9391 - dice_coef: 0.6682 - val_loss: -6.6045e-01 - val_acc:
Epoch 4/15
 - 19s - loss: -7.2053e-01 - acc: 0.9472 - dice_coef: 0.7882 - val_loss: -7.3929e-01 - val_acc:
Epoch 5/15
 - 18s - loss: -7.6174e-01 - acc: 0.9493 - dice_coef: 0.8345 - val_loss: -7.7313e-01 - val_acc:
Epoch 6/15
 - 18s - loss: -7.8110e-01 - acc: 0.9517 - dice_coef: 0.8572 - val_loss: -7.7548e-01 - val_acc:
Epoch 7/15
 - 19s - loss: -8.0069e-01 - acc: 0.9572 - dice_coef: 0.8651 - val_loss: -8.0232e-01 - val_acc:
Epoch 8/15
 - 18s - loss: -8.1617e-01 - acc: 0.9609 - dice_coef: 0.8717 - val_loss: -8.0282e-01 - val_acc:
Epoch 9/15
 - 18s - loss: -8.1364e-01 - acc: 0.9604 - dice_coef: 0.8746 - val_loss: -8.0195e-01 - val_acc:
Epoch 10/15
 - 18s - loss: -8.2044e-01 - acc: 0.9618 - dice_coef: 0.8714 - val_loss: -8.0899e-01 - val_acc:
Epoch 11/15
 - 18s - loss: -8.2184e-01 - acc: 0.9617 - dice_coef: 0.8778 - val_loss: -8.0989e-01 - val_acc:
Epoch 12/15
 - 18s - loss: -8.2939e-01 - acc: 0.9635 - dice_coef: 0.8796 - val_loss: -8.1681e-01 - val_acc:
Epoch 13/15
 - 18s - loss: -8.3228e-01 - acc: 0.9636 - dice_coef: 0.8845 - val_loss: -8.1460e-01 - val_acc:
Epoch 14/15
 - 18s - loss: -8.2932e-01 - acc: 0.9628 - dice_coef: 0.8851 - val_loss: -8.2089e-01 - val_acc:
Epoch 15/15
 - 18s - loss: -8.3372e-01 - acc: 0.9645 - dice_coef: 0.8818 - val_loss: -8.2552e-01 - val_acc:
```

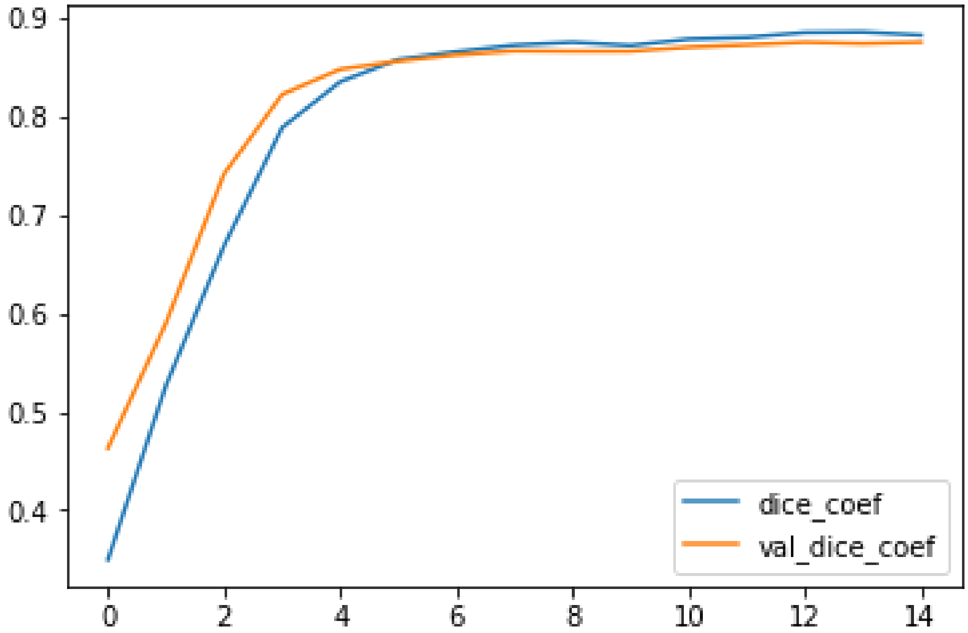
```
In [15]: plt.imshow(y_pred > 0.5, cmap='gray')
```

```
        ---------------------------------------------------------------------------
```

```
    NameError                                 Traceback (most recent call last)

    <ipython-input-15-faab2febc72c> in <module>()
----> 1 plt.imshow(y_pred > 0.5, cmap='gray')


    NameError: name 'y_pred' is not defined
```

```
In [16]: pd.DataFrame(history.history)[['dice_coef', 'val_dice_coef']].plot()
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f83c52c4cf8>
```

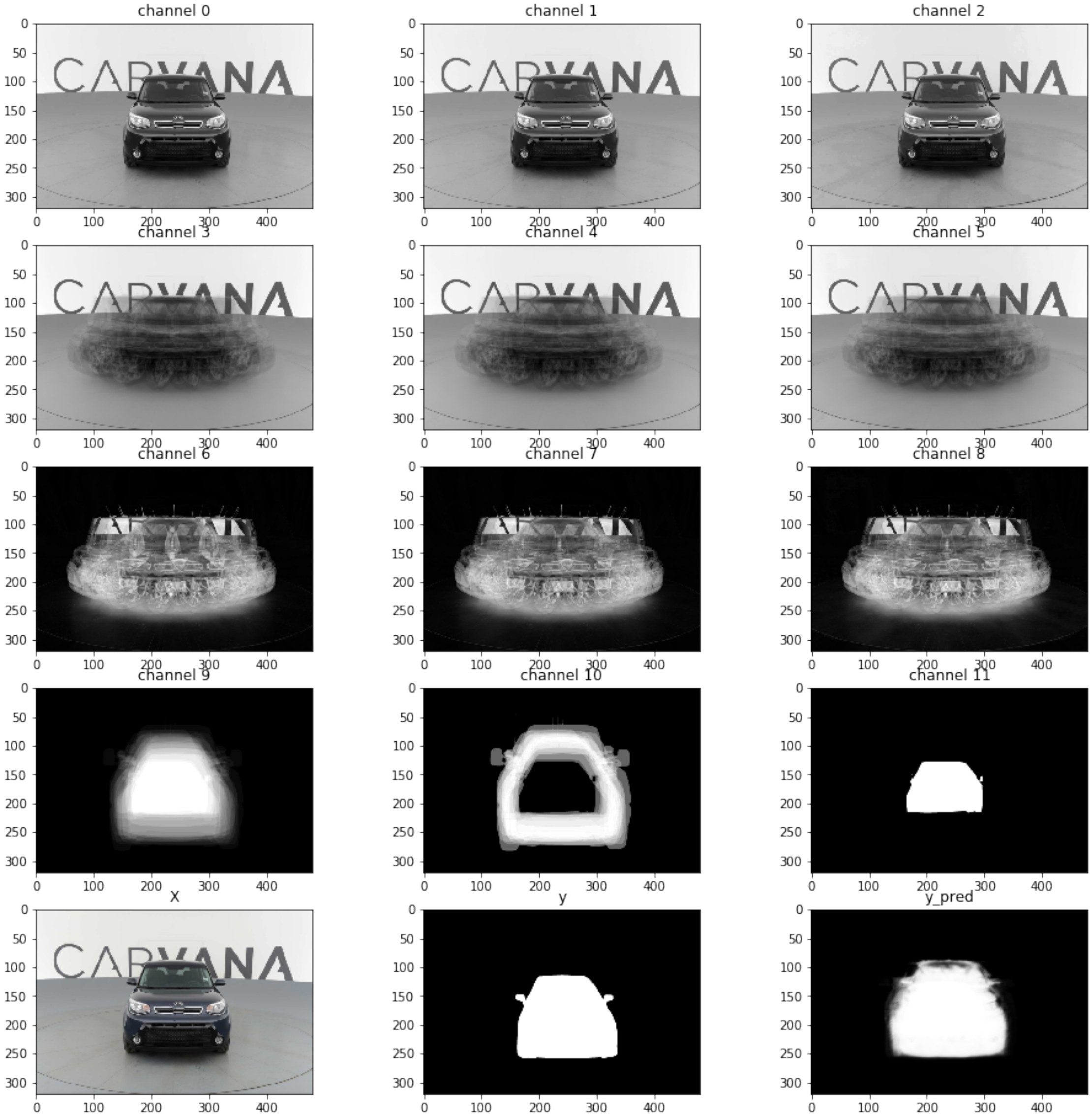


```
In [17]: idx = 0
         x = X_val[idx]

         fig, ax = plt.subplots(5,3, figsize=(16, 16))
         ax = ax.ravel()

         cmaps = ['Reds', 'Greens', 'Blues']
         for i in range(x.shape[-1]):
             ax[i].imshow(x[...,i], cmap='gray') #cmaps[i%3])
             ax[i].set_title('channel {}'.format(i))

         ax[-3].imshow((x[...,:3] * X_std[0,...,:3] + X_mean[0,...,:3]) / 255.)
         ax[-3].set_title('X')
```

```
    ax[-2].imshow(y_train[idx,...,0], cmap='gray')
    ax[-2].set_title('y')

    y_pred = model.predict(x[None]).squeeze()
    ax[-1].imshow(y_pred, cmap='gray')
    ax[-1].set_title('y_pred')
```

Out[17]:

```
Text(0.5,1,'y_pred')
```

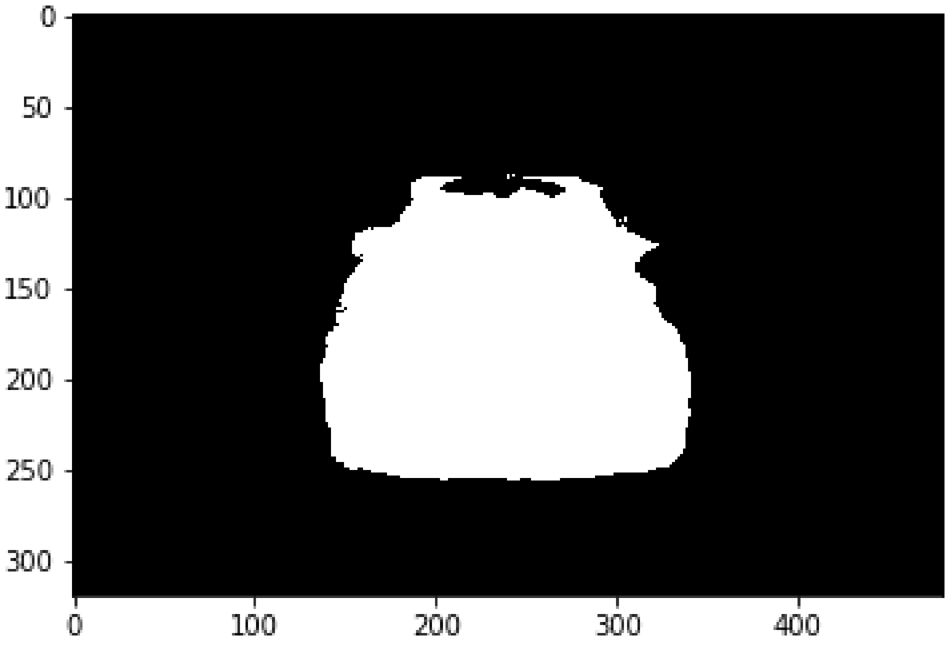


In [18]:

```
plt.imshow(y_pred > 0.5, cmap='gray')
```

Out[18]:

```
<matplotlib.image.AxesImage at 0x7f83b277d668>
```


0
50
100
150
200
250
300
0
100
200
300
400