Oxford synth text

December 12, 2018

In [1]:
```
%reload_ext autoreload
%autoreload 2
%matplotlib inline

from fastai import *
from fastai.vision import *
import coco_text
```

In [2]:
```
# for downloading dataset see: http://www.robots.ox.ac.uk/~vgg/data/scenetext

path = Path("synth-text-data/SynthText")
mat_file = path/'gt.mat'

import scipy.io
mat = scipy.io.loadmat(mat_file)
```

In [3]:
```
# get the (non-rotated) bounding boxes of one image
# actually the bounding boxes in the dataset are four points (rotated bboxes)
# in fastai the bounding boxes are non-rotated bounding boxes
def get_bboxes(img_id):
    bboxes = mat['wordBB'][0][img_id]
#     print(bboxes)
    xs = bboxes[0].transpose()
    ys = bboxes[1].transpose()
    bbxs = []
    for i in range(0,len(xs)):
        ixs = xs[i]
        iys = ys[i]
        try:
            bbxs.append( [int(min(ixs)),int(min(iys)),int(max(ixs)-min(ixs)),int(max(iy
        except:
            # Some images, eg 92 only have one bounding box, they are structured diffe
            return [[int(min(xs)), int(min(ys)), int(max(xs)-min(xs)), int(max(ys)-min
    return bbxs
```

In [4]:
```
get_bboxes(93)
```

Out[4]:
```
[[404, 263, 82, 44],
 [455, 351, 110, 58],
```

```
 [254, 183, 93, 77],
 [269, 264, 80, 49],
 [403, 238, 61, 23],
 [295, 371, 123, 39],
 [347, 12, 44, 36]]
```

In [5]:
```
get_bboxes(92)
```

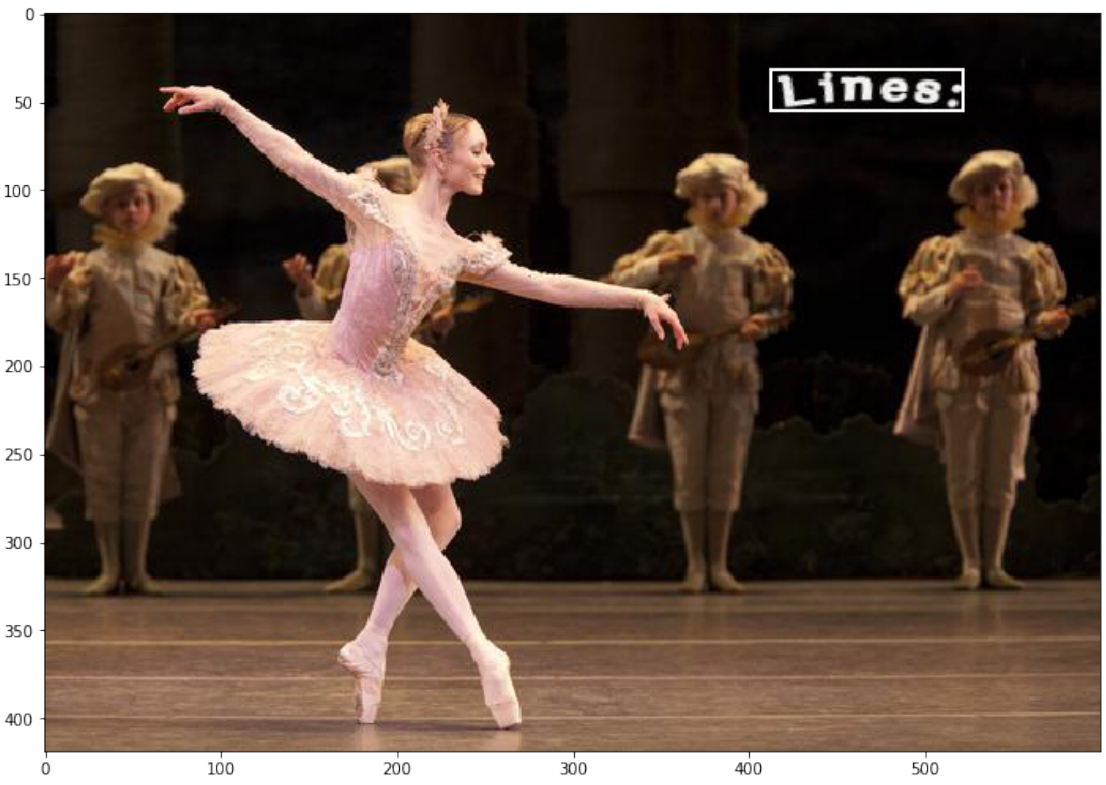
Out[5]:
```
[[412, 32, 110, 23]]
```

In [6]:
```
# Testing: show image with bounding boxes
imageId = 92
img_name = mat['imnames'][0][imageId][0]
img = open_image(path/img_name)

fig,ax = plt.subplots(figsize=(12,12))
ax.imshow(image2np(img.data))

bboxes = get_bboxes(imageId)
for bbox in bboxes:
    ax.add_patch(patches.Rectangle((bbox[0],bbox[1]),bbox[2],bbox[3], color='white', fi
ax
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f655bb62748>
```



In [7]:
```
# Testing: show an image with bounding box
imageId = 2001
img_name = mat['imnames'][0][imageId][0]
img = open_image(path/img_name)

fig,ax = plt.subplots(figsize=(12,12))
ax.imshow(image2np(img.data))

bboxes = get_bboxes(imageId)
for bbox in bboxes:
    ax.add_patch(patches.Rectangle((bbox[0],bbox[1]),bbox[2],bbox[3], color='white', fi
ax
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f655a61b6d8>
```


low
the
0
100
200
300
400
500
0
50
100
150
200
250
300
350

```
In [8]: classes = {'text': 'text'}
        annotations = []
        images = []

        for i, img in enumerate(mat['imnames'][0]):
            images.append( {'id': i, 'file_name': img[0]} )

        for i, img in enumerate(mat['imnames'][0]):
            bboxes = get_bboxes(i)
            for bbox in bboxes:
                annotations.append({'bbox': bbox, 'image_id': i, 'category': 'text'})

        annot_dict = {'annotations': annotations, 'images': images, 'classes': classes}

In [9]: def get_annotations(annot_dict, prefix=None):
            id2images, id2bboxes, id2cats = {}, collections.defaultdict(list), collections.defa

            classes = annot_dict['classes']
            for o in annot_dict['annotations']:
                bb = o['bbox']
                id2bboxes[o['image_id']].append([bb[1],bb[0], bb[3]+bb[1], bb[2]+bb[0]])
                id2cats[o['image_id']].append(classes[o['category']])
            for o in annot_dict['images']:
                if o['id'] in id2bboxes:
        #             print(o)
                    id2images[o['id']] = ifnone(prefix, '') + o['file_name']
            ids = list(id2images.keys())
            return [id2images[k] for k in ids], [[id2bboxes[k], id2cats[k]] for k in ids]

In [10]: images, lbl_bbox = get_annotations(annot_dict)
         img2bbox = dict(zip(images, lbl_bbox))

In [11]: images = [i[0] for i in mat['imnames'][0]]
         img_df = pd.DataFrame(images, columns=['file_name'])
         img_df.head()

Out[11]:              file_name
         0    8/ballet_106_0.jpg
         1    8/ballet_106_1.jpg
         2   8/ballet_106_10.jpg
         3  8/ballet_106_100.jpg
         4  8/ballet_106_101.jpg

In [12]: def get_y_func(o):
             index = o.split('/')[-2]+"/"+o.split('/')[-1]
             return img2bbox[index]

In [13]: annot_df = pd.DataFrame(lbl_bbox, columns=['bbox', 'classes'])
         annot_df.head()
```

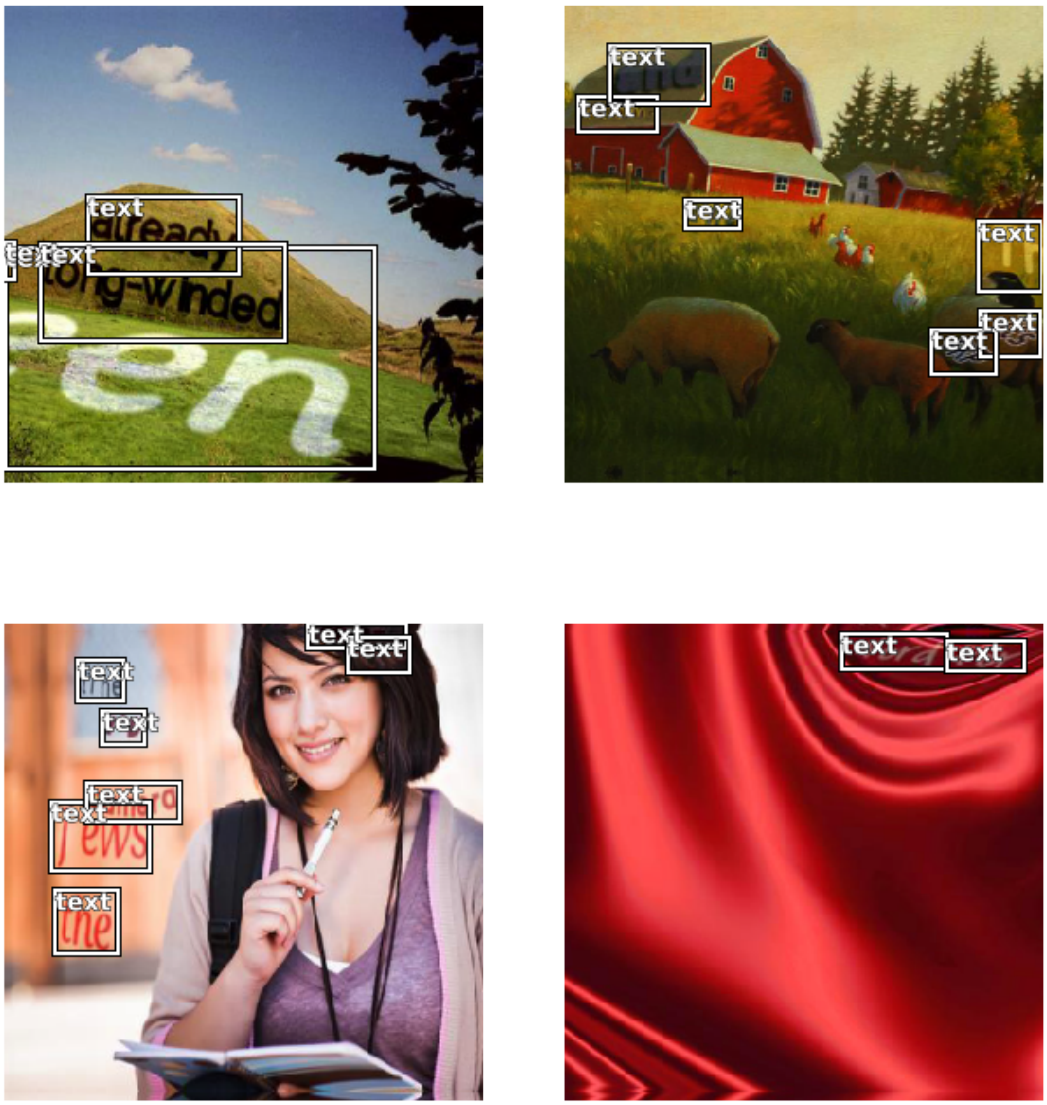
Out[13]:

	bbox	classes
0	[[21, 420, 41, 512], [46, 415, 68, 431], [47, ...	[text, text, text, text, text, text, text, tex...
1	[[231, 255, 299, 328], [14, 424, 32, 533], [39...	[text, text, text, text, text, text, text, tex...
2	[[47, 322, 74, 383], [352, 430, 397, 572], [19...	[text, text, text, text, text, text, text, tex...
3	[[213, 257, 294, 331], [6, 427, 30, 552], [37,...	[text, text, text, text, text, text, text, text]
4	[[200, 252, 253, 313], [269, 253, 318, 319], [...	[text, text, text, text, text, text, text, tex...

In [14]:
```
path = Path('/home/jupyter/coco')

data = (ObjectItemList.from_df(img_df, path, folder='synth-text-data/SynthText')
        #Where are the images? -> in coco
        .random_split_by_pct()
        #How to split in train/valid? -> randomly with the default 20% in valid
        .label_from_func(get_y_func)
        #How to find the labels? -> use get_y_func
        .transform(get_transforms(do_flip=False, flip_vert=False), size=300, tfm_y=Tru
        #Data augmentation? -> Standard transforms with tfm_y=True
        .databunch(bs=20, collate_fn=bb_pad_collate))
        #Finally we convert to a DataBunch and we use bb_pad_collate
```

In [15]:
```
data.show_batch(2)
```




```
In [16]: learn = create_cnn(data, models.resnet50, metrics=error_rate, bn_final=True)

In [17]: learn.fit_one_cycle(2)

<IPython.core.display.HTML object>
```

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

ValueError                                Traceback (most recent call last)

<ipython-input-17-390c4543c0c9> in <module>
```

```
----> 1 learn.fit_one_cycle(2)


    /opt/anaconda3/lib/python3.6/site-packages/fastai/train.py in fit_one_cycle(learn, cyc
     18     callbacks.append(OneCycleScheduler(learn, max_lr, moms=moms, div_factor=div_fa
     19                                        pct_start=pct_start, **kwargs))
---> 20     learn.fit(cyc_len, max_lr, wd=wd, callbacks=callbacks)
     21
     22 def lr_find(learn:Learner, start_lr:Floats=1e-7, end_lr:Floats=10, num_it:int=100,


    /opt/anaconda3/lib/python3.6/site-packages/fastai/basic_train.py in fit(self, epochs, l
    160         callbacks = [cb(self) for cb in self.callback_fns] + listify(callbacks)
    161         fit(epochs, self.model, self.loss_func, opt=self.opt, data=self.data, metri
--> 162             callbacks=self.callbacks+callbacks)
    163
    164     def create_opt(self, lr:Floats, wd:Floats=0.)->None:


    /opt/anaconda3/lib/python3.6/site-packages/fastai/basic_train.py in fit(epochs, model,
     92     except Exception as e:
     93         exception = e
---> 94         raise e
     95     finally: cb_handler.on_train_end(exception)
     96


    /opt/anaconda3/lib/python3.6/site-packages/fastai/basic_train.py in fit(epochs, model,
     82             for xb,yb in progress_bar(data.train_dl, parent=pbar):
     83                 xb, yb = cb_handler.on_batch_begin(xb, yb)
---> 84                 loss = loss_batch(model, xb, yb, loss_func, opt, cb_handler)
     85                 if cb_handler.on_batch_end(loss): break
     86


    /opt/anaconda3/lib/python3.6/site-packages/fastai/basic_train.py in loss_batch(model, x
     20
     21     if not loss_func: return to_detach(out), yb[0].detach()
---> 22     loss = loss_func(out, *yb)
     23
     24     if opt is not None:


    /opt/anaconda3/lib/python3.6/site-packages/torch/nn/functional.py in binary_cross_entro
   1768
   1769     if not (target.size() == input.size()):
-> 1770         raise ValueError("Target size ({}) must be the same as input size ({})".fo
   1771
```

```
1772     return torch.binary_cross_entropy_with_logits(input, target, weight, pos_weight
```

```
ValueError: Target size (torch.Size([20, 12, 4])) must be the same as input size (torch
```

In [ ]: