Deep Learning and Practice
Lab 7: Caption generation with visual attention

李韡
0556157
fm.bigballon@gmail.com

Introduction

In this Lab, we are going to run a Caption generator by using CNN and RNN language generator to generate a sentence that describes the image.
We will modify the code (https://github.com/yunjey/show-attend-and-tell)

- Upgrade code for tensorflow 1.0 (model.py, solver.py).
- Deal with the memory problem

Lab Description:

- Learn how to combine CNN features and RNN language generator.
- Compare two different attention mechanisms.

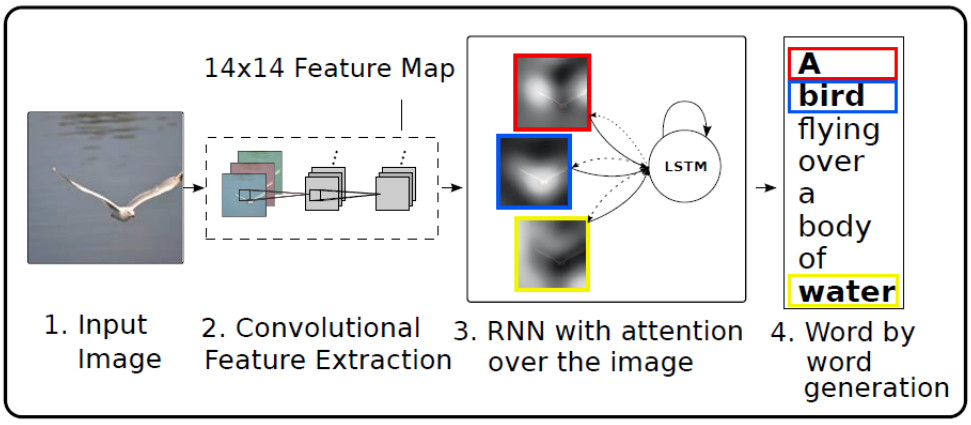


Figure 1: Structure of model

Experiment setup

- Upgrade code for tensorflow 1.0 (model.py, solver.py).

1. Download tf_upgrade.py (https://github.com/tensorflow/tensorflow/tree/master/tensorflow/tools/compatibility)
2. Using **[python tf_upgrade.py --infile InputFile --outfile OutputFile]** **to update**

- Deal with the memory problem

1. prepro.py main ()

Because our machine has only 12GB memory, so we can't load all train data. So I split features to 9 files (10000 each, the last one is 2783).

And to run this code, I change the **batch_size** from 100 to 40

```
# split features.hkl to 9 files
# print n_examples
# 82783 / 10000 -> 9

part_size = 10000
for size in range(int(math.ceil(n_examples / 10000.0))):
    save_path = './data/%s/%s.features%d.hkl' % (split, split, size)

    # 10000 features in each part
    # the last part is 2783 features
    st = size * 10000
    ed = (size + 1) * 10000
    if ed > n_examples:
        ed = n_examples
    cur_size = ed - st
```

2. utils.py

If [split == 'train'], we will not load features.hkl.

```
    # loading features in training time
    if split != 'train':
        data['features'] = hickle.load(os.path.join(data_path, '%s.features.hkl' %split))
```

3. solver.py

In function train (), each epoch, we will load the 9 feature files in turn.

```
        for e in range(self.n_epochs):
            # load data 9 times to solve huge data load problem #
            cur_iteration = 0
            for data_cnt in range(9):
                print "-------------------------------------------------"
                print "Loading data (part %d / 9) " %(int(data_cnt)+1)

                features = hickle.load(os.path.join('./data/train', 'train.features%d.hkl' % data_cnt))

                total_num = features.shape[0]
                print "Load success (data size: %d) " %total_num
                print "Iterations: %d" %n_iters_per_epoch
                print "-------------------------------------------------"
```

4. train.py

```
# solver.train()
solver.test(val_data, split='val')
```

5. model.py

change softmax alpha matrix to **one hat matrix** (argmax - 1 others - 0)

```
# alpha = tf.nn.softmax(out_att)
# using argmax -> set max to 1   others to 0
alpha = tf.one_hot(tf.argmax(out_att, axis=1), self.L, on_value=1.0, off_value=0.0, axis=-1)
context = tf.reduce_sum(features * tf.expand_dims(alpha, 2), 1, name='context')
return context, alpha
```

Result

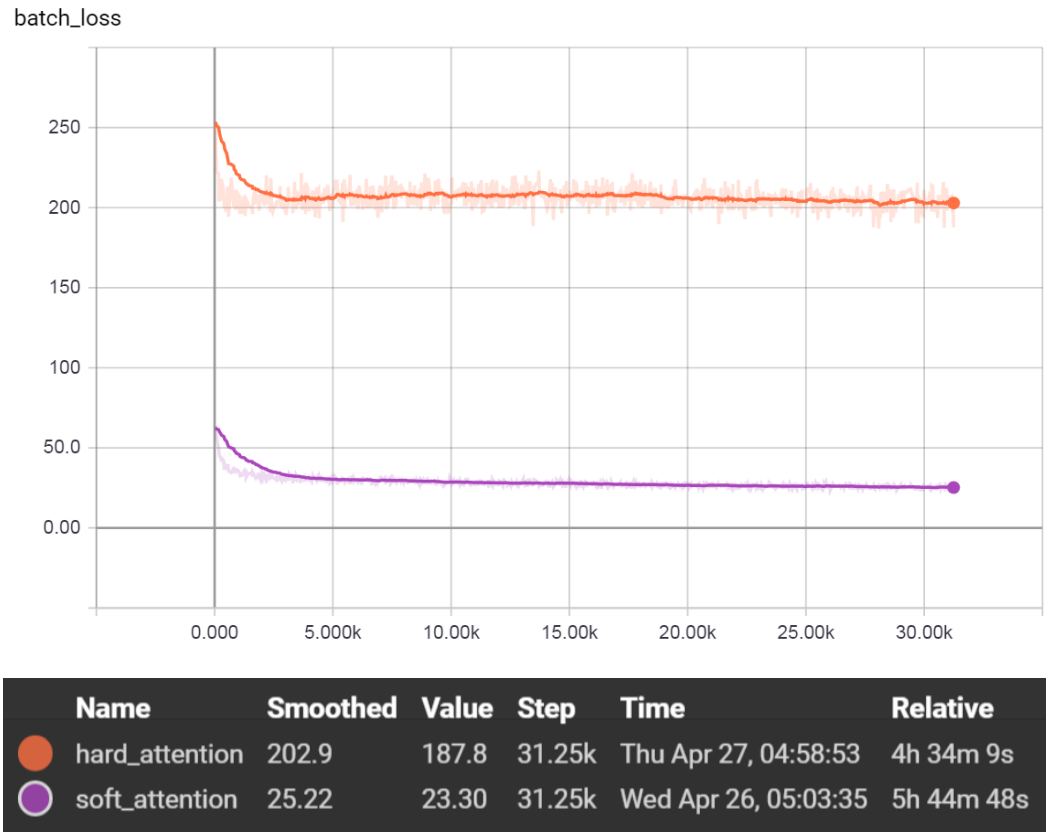
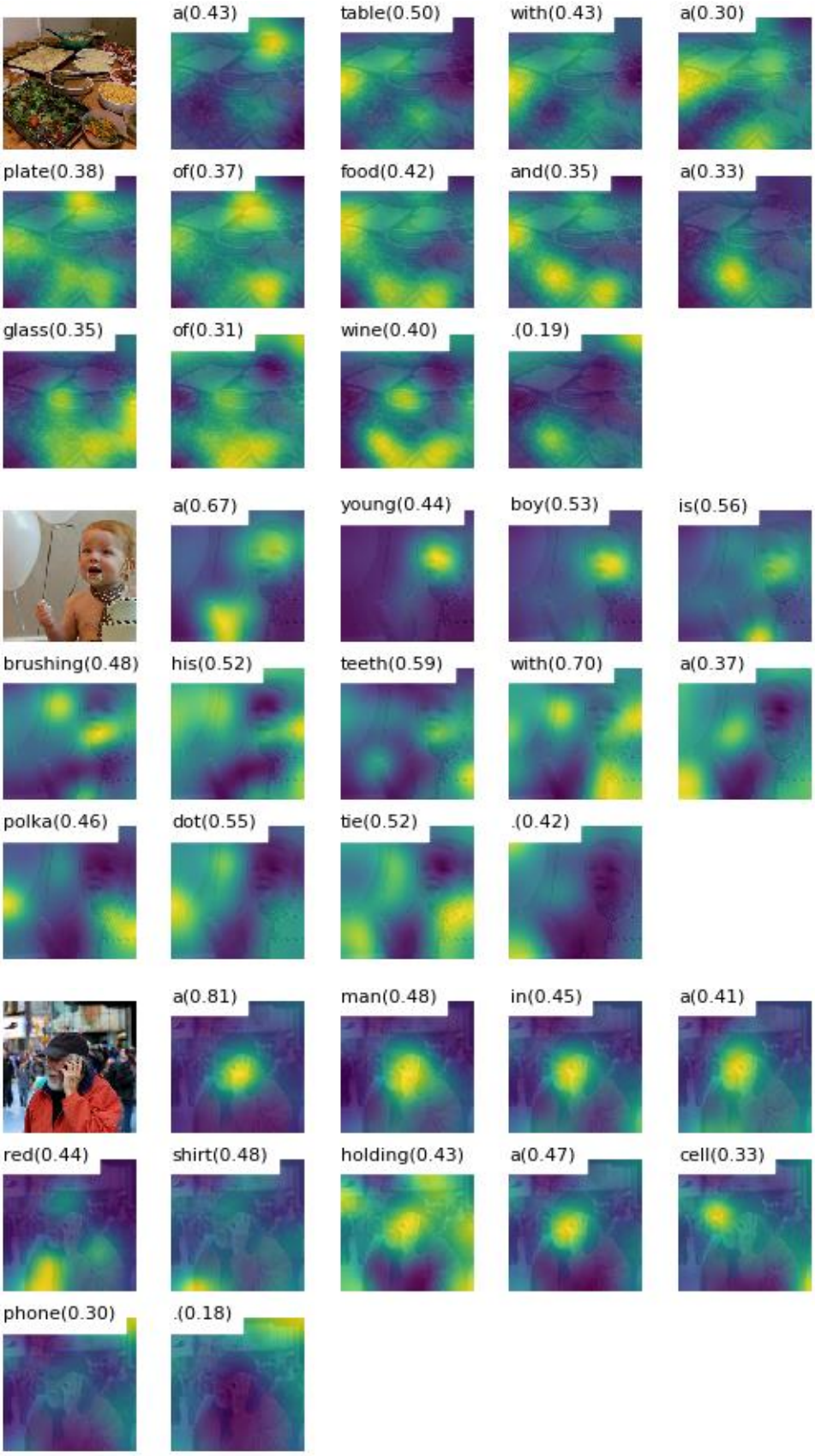
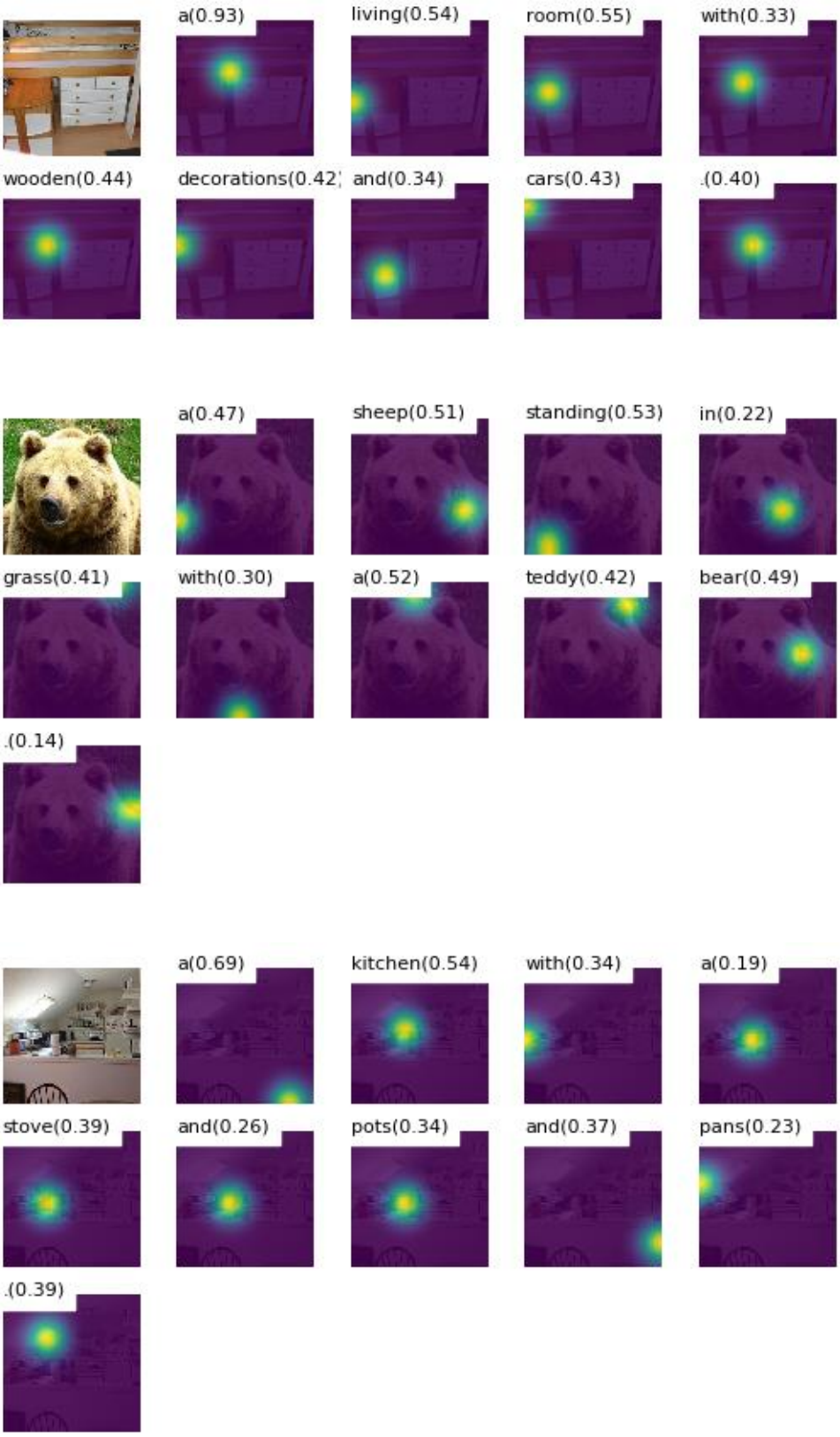


Figure 2: batch loss of hard/soft attention

- **Some images:**
  - Soft attention:



- Hard attention (using one hat vector):



Discussion

About hard attention image:

Dataset	Model	BLEU				METEOR
		BLEU-1	BLEU-2	BLEU-3	BLEU-4	
Flickr8k	Google NIC(Vinyals et al., 2014)†Σ	63	41	27	—	—
	Log Bilinear (Kiros et al., 2014a)°	65.6	42.4	27.7	17.7	17.31
	Soft-Attention	**67**	44.8	29.9	19.5	18.93
	Hard-Attention	**67**	**45.7**	**31.4**	**21.3**	**20.30**
Flickr30k	Google NIC†°Σ	66.3	42.3	27.7	18.3	—
	Log Bilinear	60.0	38	25.4	17.1	16.88
	Soft-Attention	66.7	43.4	28.8	19.1	**18.49**
	Hard-Attention	**66.9**	**43.9**	**29.6**	**19.9**	18.46
COCO	CMU/MS Research (Chen & Zitnick, 2014)a	—	—	—	—	20.41
	MS Research (Fang et al., 2014)†a	—	—	—	—	20.71
	BRNN (Karpathy & Li, 2014)°	64.2	45.1	30.4	20.3	—
	Google NIC†°Σ	66.6	46.1	32.9	24.6	—
	Log Bilinear°	70.8	48.9	34.4	24.3	20.03
	Soft-Attention	70.7	49.2	34.4	24.3	**23.90**
	Hard-Attention	**71.8**	**50.4**	**35.7**	**25.0**	23.04

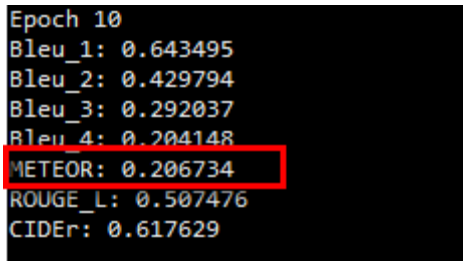
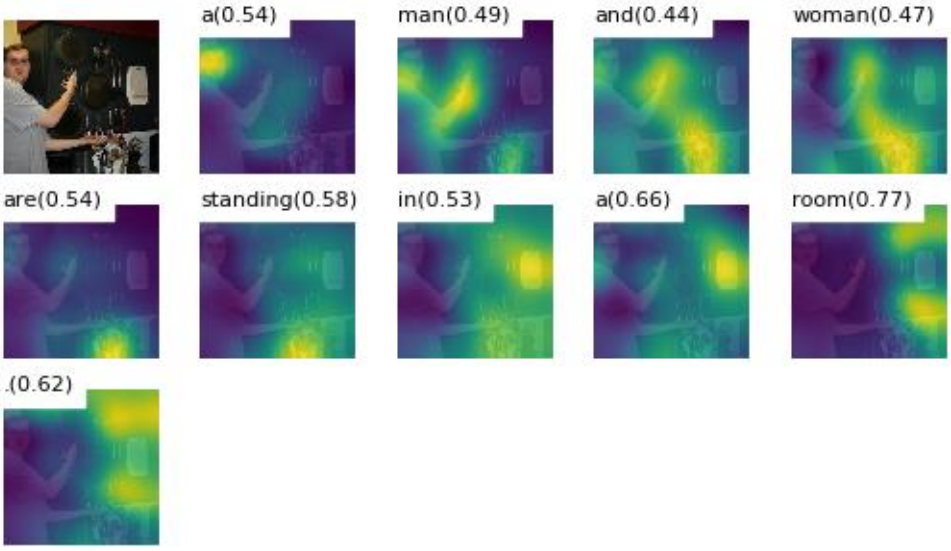


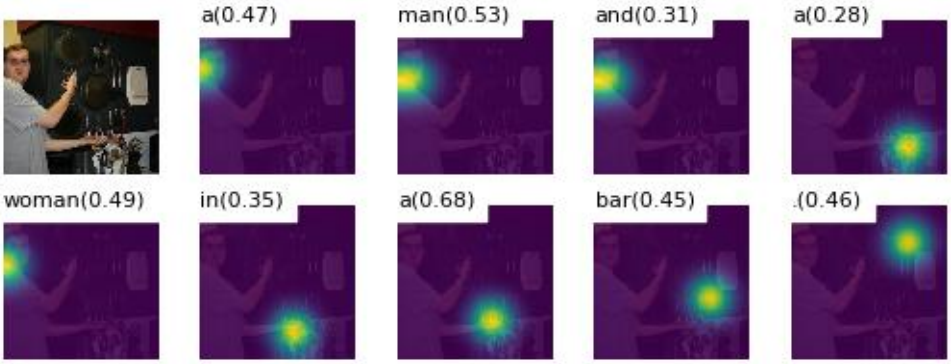
Figure 3: val.bleu.scores.txt of hard attention

The hard attention's performance is worse than soft attention. (see figure 2's batch loss)
According to paper, the METEOR is about 23%(training by sampling)
in my lab (10 epochs, about 5 hour's training, using one hat vector), I got 20.6%.

From figure 4 & figure 5, we also can see after 10 epoch's training, my can got some good captions.
But in figure 6, maybe hard attention's caption is incorrect?

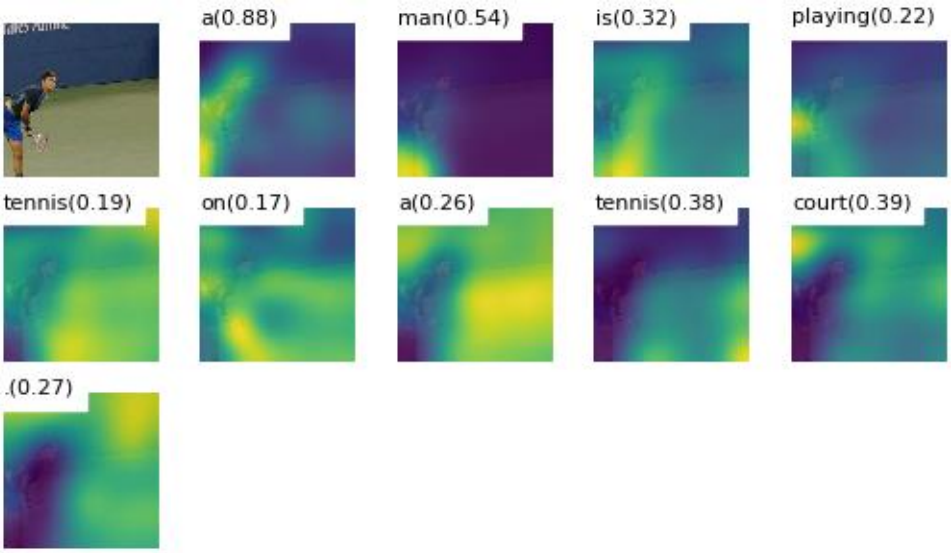


(a) A man and woman are standing in a room.

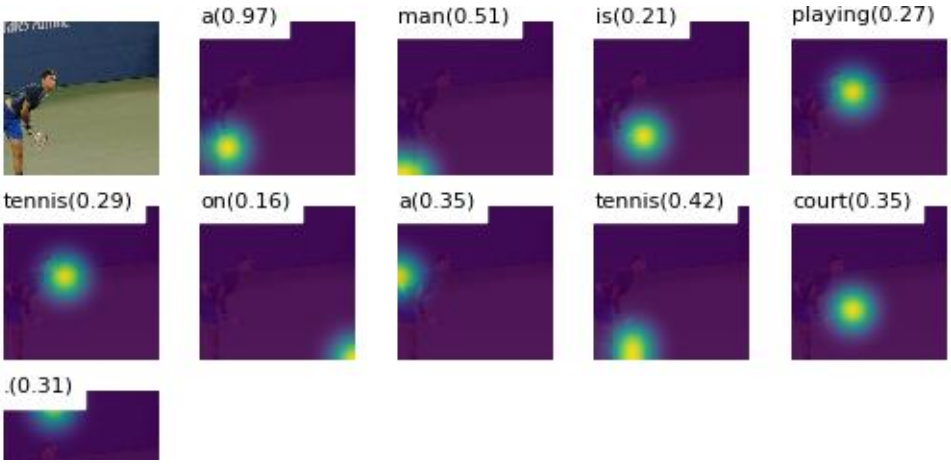


(b) A man and a woman in a bar.

Figure 4: sample 1 of soft (a)/ hard(b) attention

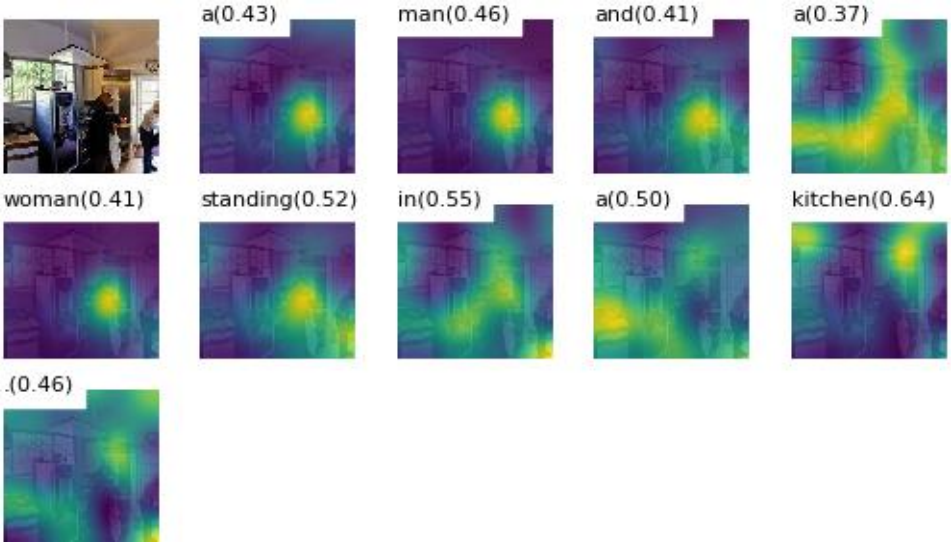


(a) A man is playing tennis on a tennis court.

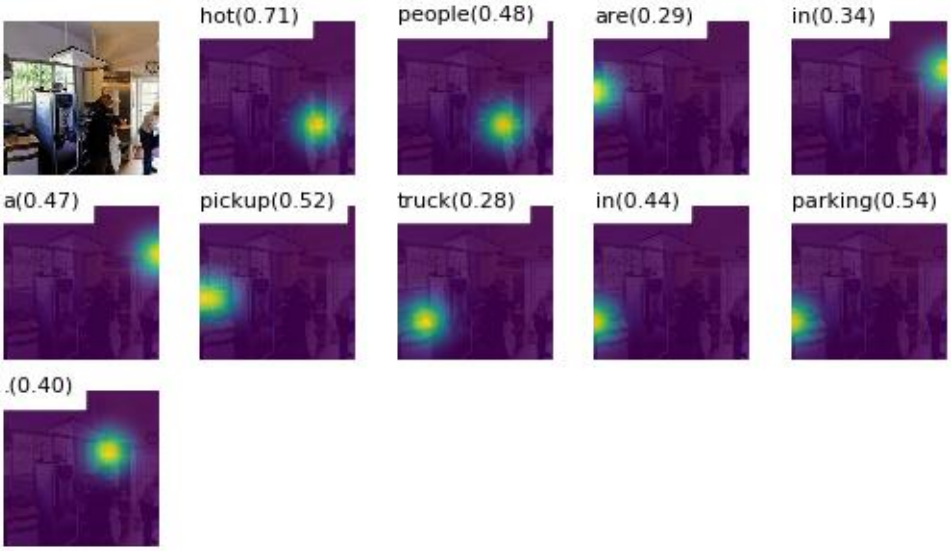


(b) A man is playing tennis on a tennis court.

Figure 5: sample 2 of soft (a)/ hard(b) attention



(a) A man and a woman standing in a kitchen.



(b) Hot people are in a pickup truck in parking.

Figure 6: sample 3 of soft (a)/ hard(b) attention