COGS 260 Spring 2018: Assignment 4

Qiaojun Feng
University of California, San Diego
qjfeng@ucsd.edu

Abstract

In this assignment we are working on the following questions:

1. *Feed-forward Neural Network*
2. *Char RNN*
3. *Object Detection*

1. Method(Derivation)

1.1. Feed-forward Neural Network

First we apply the Chain Rule to derive the partial derivative.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$
$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$$
$$f(x) = \sigma(w_2^T(\sigma(w_1^T x + b_1)) + b_2)$$

$$L_i = -(y^{(i)} \ln f(x_i) + (1-y^{(i)}) \ln(1-f(x_i)))$$
$$L = -\sum_i (y^{(i)} \ln f(x_i) + (1-y^{(i)}) \ln(1-f(x_i))) = \sum_i L_i$$

$$\frac{\partial L_i}{\partial w_2} = (f(x_i) - y_i)\sigma(w_1^T x + b_1)$$
$$\frac{\partial L}{\partial w_2} = \sum_i (f(x_i) - y_i)\sigma(w_1^T x + b_1)$$

$$\frac{\partial L_i}{\partial w_1}$$
$$= \frac{\partial L_i}{\partial f(x)} \frac{\partial f(x)}{\partial(\sigma(w_1^T x + b_1))} \frac{\partial(\sigma(w_1^T x + b_1))}{\partial w_1}$$
$$= x_i \cdot [(f(x_i) - y_i) \cdot w_2 \cdot \sigma(w_1^T x + b_1) \cdot (1 - \sigma(w_1^T x + b_1))]^T$$
$$\frac{\partial L}{\partial w_1}$$
$$= \sum_i x_i \cdot [(f(x_i) - y_i) \cdot w_2 \cdot \sigma(w_1^T x + b_1) \cdot (1 - \sigma(w_1^T x + b_1))]^T$$

Above is the derivation of the partial derivative w.r.t w_1, w_2. Apply gradient descent to decrease the loss.

1.2. Char RNN

Since we are generate the characters, we enumerate all existing characters(including space, new line and punctuation while case sensitive). Suppose there are n characters in total, then we use one hot encoding to encode each character as a vector $x \in \mathbb{R}^n$. Likewise, the output is also an n-d vector with sum of all n elements to be one. Using a softmax layer before the output can satisfy such requirement.

We use the implementation from [2]. The network architecture is shown in Figure 1. Each LSTM cell is connected with the upper layer and lower layer's cell, as well as the temporal-following cell. In each LSTM cell the number of unit is fixed as m, i.e., an embedded feature vector. Finally the softmax layer maps the m-d vector to $\mathbb{R}^n$.

1.3. Object Detection

For object detection, we want to test the model's ability for Scale Invariance and Rotation Invariance. We use the implementation from [1].

2. Experiments

2.1. Feed-forward Neural Network

Here we plot the training loss vs. the number of iterations and the training error and test error vs. the number of iterations (Figure 2). The final training error is 0 and test error is 0.133.

2.2. Char RNN

The hyperparameters we use are list below in Table 1.

And we report the training loss in Figure 3.

We can show part of the generated contents but unfortunately it is not easy to understand. However, it learns to some words and learns to separate the scripts and the role's name on different lines at least.

Start with "All":

All the dead can tell me her, And I will say you say you would have stay to be A son of thy tongue from me to the

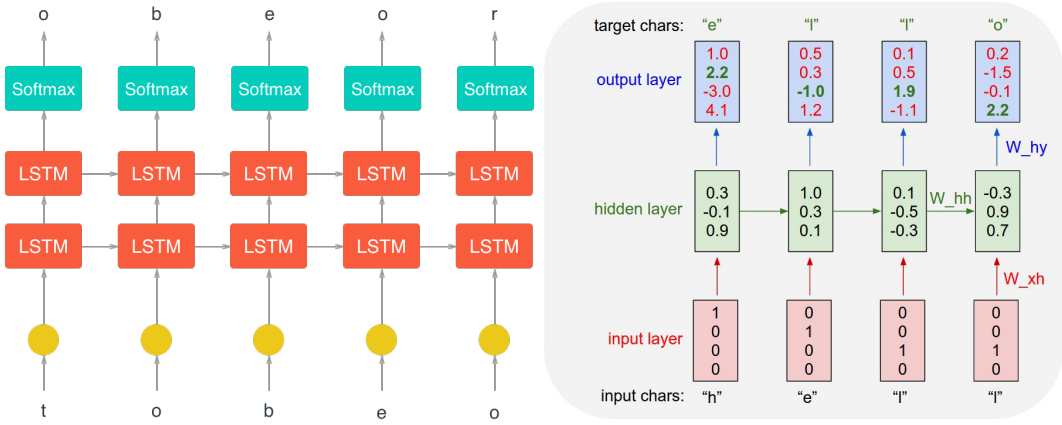

Figure 1. LSTM network architecture

state, To speak at her here she may be to thee.

DUCHESS OF YORK: When they have stands along with him. And, both Will not be to my soul in the dear man: By his sword stops to have a will of step And beat the second clust than this to-morrow.

PRINCE EDWARD: I have some cape there with your harm'd that will The precious dravent of his crown of the measure, Though I have breed imporsing on my heart Where you have been to have a father's head. And what say you? and so I would he speaks That we will both is this the mundard off, To show me all as mently with a fire Would never speak: what are you? here is there to make A provost that the starp would bear that mother, Before I do so married to the corns and hide. And since thou shalt be the death woe to hurt.

Start with "Lord":

Lord Capulet!

Network architecture	
num_LSTM_layers	2
num_LSTM_units	512
Batch setting	
batch_size	100
sequence_size	100
Training setting	
learning_rate	0.001
dropout_p	0.6
n_epochs	70

Table 1. Hyperparameters setting

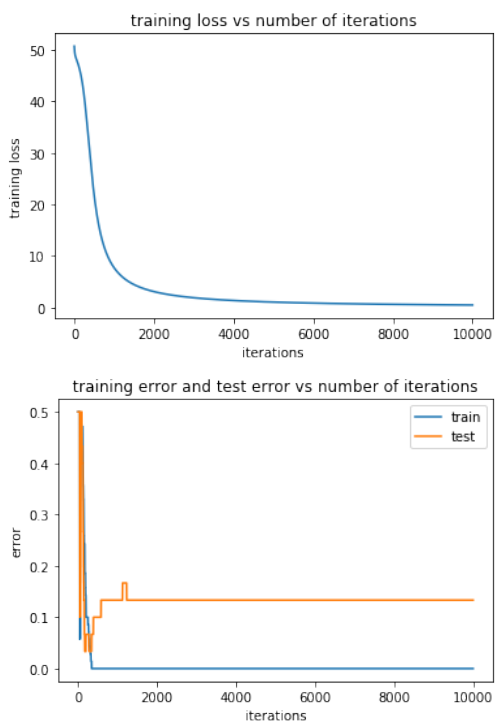

Figure 2. loss / error for feed-forward neural network

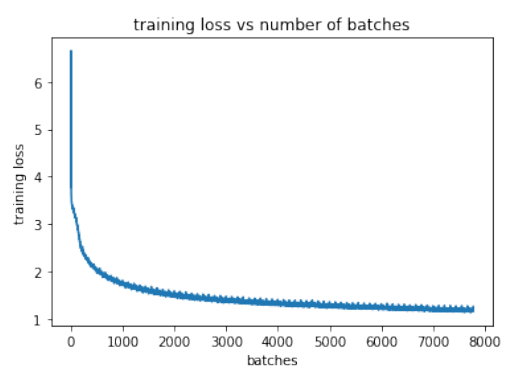

Figure 3. training loss for Char RNN

KING LEWIS XI: The shepherd speaks well all, or shorted way, The cause of this care of the provest tongue. Who deness at her and mother there the man?

LADY CAPULET: That is a winder side.

More details can be viewed in the notebook file.

2.3. Object Detection

We test 4 different pre-trained model: SSD, RFCN, Faster R-CNN and Mask R-CNN. We pick some results and show in Figure 4 5 6 7.

For the scale testing, we try $4 \times 4, 8 \times 8, 16 \times 16$ and keep the same size of $4n \times 4m$, where $n \times m$ is the origin image size. So when the images repeat more, each of them is smaller.

For the rotation testing, we rotate with 30° interval and keep the same size as origin image. The outside pixels are cropped and the empty space is padded with 0(black pixels).

3. Discussion

We mainly focus on object detection and discuss what might cause the detection results when there are variations in scale and direction. The core idea here is that the ability of the model is somehow the property of the training dataset, because we can always view the model as a function of the training set. Sometimes the restriction of the model is due to the limitation of the training set.

3.1. Scale Invariance

Actually we also try 8×8 and 16×16 image collage but there are seldom detection results. For some categories like cat and dog we show, it can be detected. However for some other categories like airplane, laptop, the smaller object is not easy to be detected.

One possible explanation is that the detectors for those categories have some scale priori condition, that is in the training set those categories are always annotated as similar scale, while smaller scale annotations are lacked. The other possible reason is that the image collages break the spatial configuration of natural image since it is actually artificial image, but the detector is trained on mostly natural images.

3.2. Rotational Invariance

Same as the hypothesis above, the rotation breaks the spatial configuration of natural image so detector becomes insensitive. However, objects that are small and are easy to rotate are easy to be detected under the rotated configuration. For example in Figure 7 the yellow boxes are for kites actually. We can see it is relative robust compared with the human detection. It can be explained that we seldom have image with people upside-down, but the kite might rotate any angle in the sky which can be caught in the training set.

The best solution is to have more and more annotated training dataset, but obviously it is too expensive. One possible alternative method is to augment the existing training set, which is popular preprocessing method. We believe these models have tried data augmentation, but still for some artifact the detector degenerates severely. This reminds us that it is never easy to learn a real object concept from the training data efficiently.

References

[1] tensorflow. models/research/object_detection. https://github.com/tensorflow/models/tree/master/research/object_detection. 1

[2] udacity. deep-learning/anna_karnna_solution.ipynb. https://github.com/udacity/deep-learning/blob/master/intro-to-rnns/Anna_KaRNNa_Solution.ipynb. 1

Figure 4. detection on 4×4 image collage from left to right: SSD, RFCN, Faster RCNN, Mask RCNN

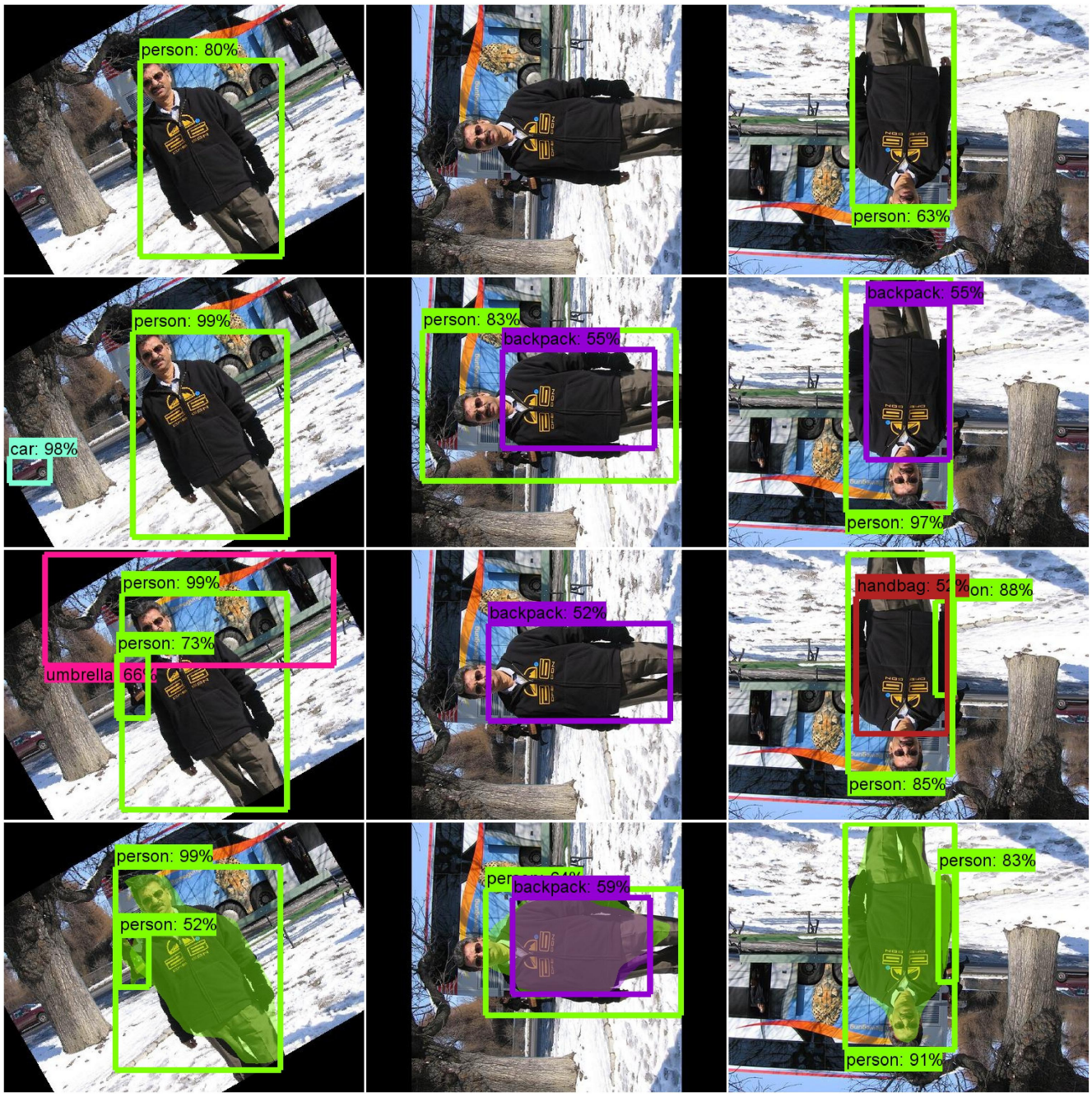

Figure 5. detection on $30^\circ, 90^\circ, 180^\circ$ image collage from up to down: SSD, RFCN, Faster RCNN, Mask RCNN

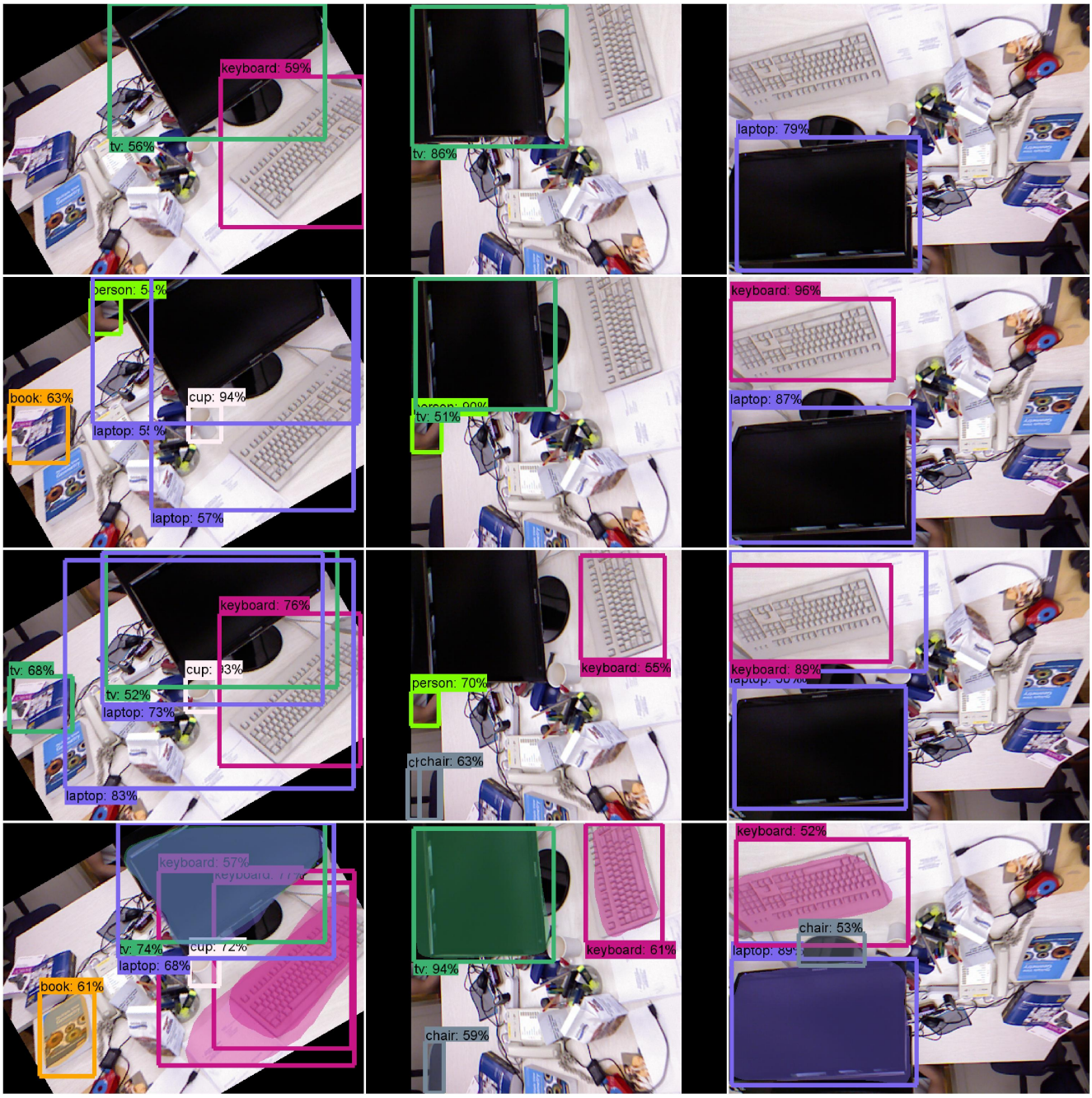

Figure 6. detection on $30^\circ, 90^\circ, 180^\circ$ image collage from top to down: SSD, RFCN, Faster RCNN, Mask RCNN

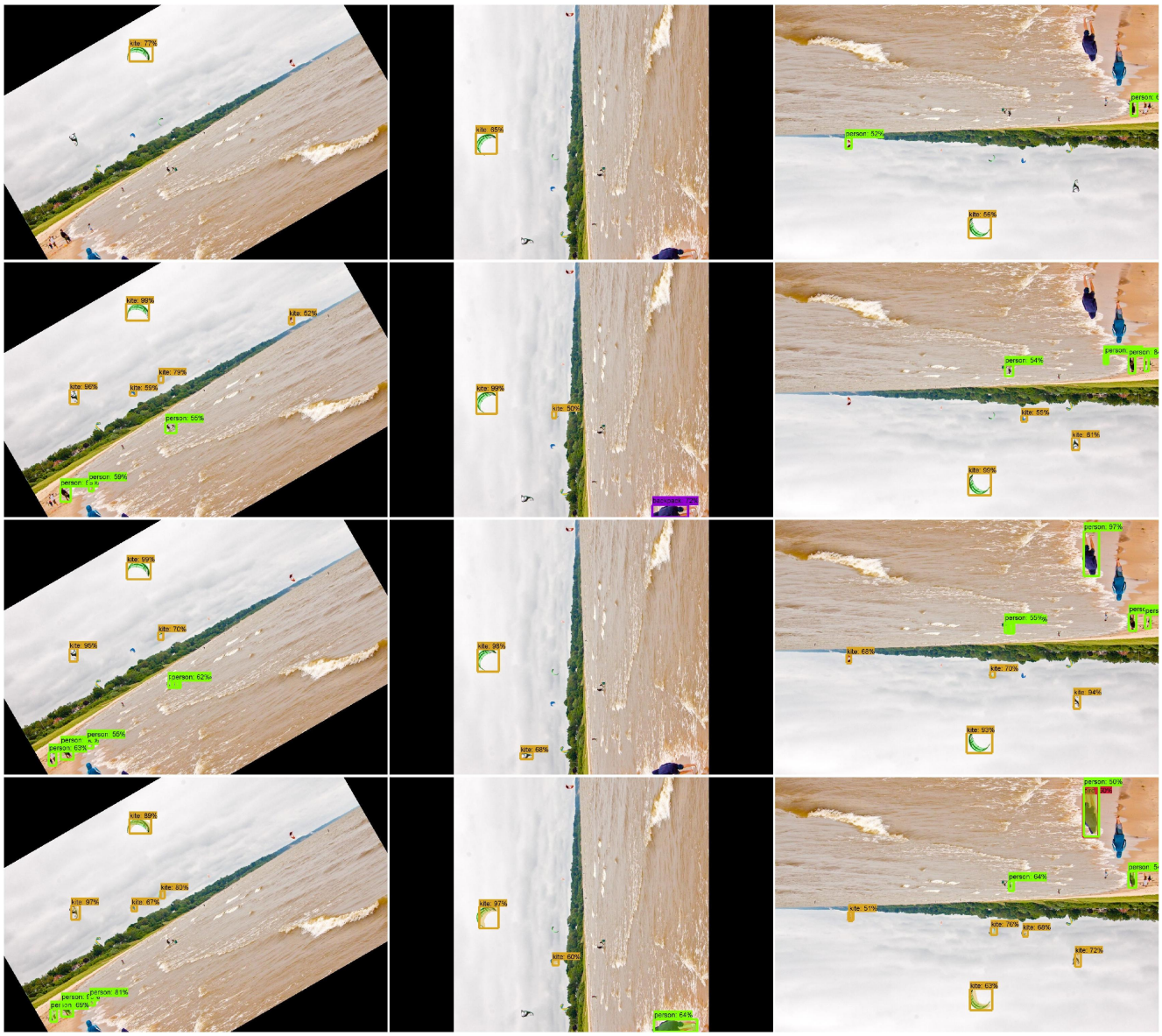

Figure 7. detection on $30^\circ, 90^\circ, 180^\circ$ image collage from top to down: SSD, RFCN, Faster RCNN, Mask RCNN