Focal loss with multi-label implemented in keras

Focal loss

- focal loss with multi-label implemented in keras.
- reference to paper : Focal Loss for Dense Object Detection
- add LSR (label smoothing regularization)

Usage

- firstly, you should get a list which contains each class number, like classes_nu=[1,2,3] means index_0 class have 1 pic, index_1 class have 1 pics, index_2 class have 3 pics.
- then, use the focal loss function like below:

```
model.compile(optimizer=SGD(lr=learning_rate, momentum=0.9), loss=
[focal_loss(classes_num)], metrics=['accuracy'])

  # focal loss with multi label
  def focal_loss(classes_num, gamma=2., alpha=.25, e=0.1):
      # classes_num contains sample number of each classes
      def focal_loss_fixed(target_tensor, prediction_tensor):
          '''
          prediction_tensor is the output tensor with shape [None, 100], where
  100 is the number of classes
          target_tensor is the label tensor, same shape as predcition_tensor
          '''
          import tensorflow as tf
          from tensorflow.python.ops import array_ops
          from keras import backend as K

          #1# get focal loss with no balanced weight which presented in paper
  function (4)
          zeros = array_ops.zeros_like(prediction_tensor,
  dtype=prediction_tensor.dtype)
          one_minus_p = array_ops.where(tf.greater(target_tensor,zeros),
  target_tensor - prediction_tensor, zeros)
          FT = -1 * (one_minus_p ** gamma) *
  tf.log(tf.clip_by_value(prediction_tensor, 1e-8, 1.0))

          #2# get balanced weight alpha
          classes_weight = array_ops.zeros_like(prediction_tensor,
  dtype=prediction_tensor.dtype)

          total_num = float(sum(classes_num))
          classes_w_t1 = [ total_num / ff for ff in classes_num ]
          sum_ = sum(classes_w_t1)
          classes_w_t2 = [ ff/sum_ for ff in classes_w_t1 ]   #scale
          classes_w_tensor = tf.convert_to_tensor(classes_w_t2,
  dtype=prediction_tensor.dtype)
          classes_weight += classes_w_tensor

          alpha = array_ops.where(tf.greater(target_tensor, zeros),
  classes_weight, zeros)

          #3# get balanced focal loss
          balanced_fl = alpha * FT
          balanced_fl = tf.reduce_mean(balanced_fl)

          #4# add other op to prevent overfit
          # reference : https://spaces.ac.cn/archives/4493
          nb_classes = len(classes_num)
          fianal_loss = (1-e) * balanced_fl + e *
  K.categorical_crossentropy(K.ones_like(prediction_tensor)/nb_classes,
  prediction_tensor)

          return fianal_loss
      return focal_loss_fixed
```

Code