Homework5

Name: Yang Cai
NUID: 001632759

Pseudo code

```
// imageHashMap maps (imageId, layerNum) to a layer of pixel
imageHashMap = (image0, image1…image5)
                        .map(image => read(image)
                                // change key from layerNum to (imageId, layerNum)
                                // so that we can differentiate between imgaes
                                .map((layerNum, layerPixel) => (image, layerNum), layerPixel))
                        .reduce(_ ++ _)
broadcast(imageHashMap)

// read *_dist.tiff files, convert distance value to label, and for each layer, generate
/ ((imageId, layerNum), layerLabel), filter out the edges on the z dimension
distances = (image0, image1…image5)
                        .flatMap{image => read(image)
                                // change key from layer to (imageId, layer)
                                // so that we can differentiate between imgaes
                                .map((layer, layerDists) => ((image, layer), label(layerDists)))
                                // filter out z-axis's edges
                                .filter(remove_z_dim_edges)

// res is sample of 12500 points, each point contains its label and neighborhoods array
val res = sparkContext.parallize(distances, 200).flatMap{
        // for each layer, generate list of (label, neighborhoods) for each pixel on this layer
        ((imageId, layerNum), layerDists) =>
        layerDists.zipWithIndex    // generate (label, position) for each point on this layer
                .filter(remove_unknow_pixel and remove x,y_dim_edges)
                // position is the point's position on this layer
                .map((label, position) =>
                                val neighborhood = getNeighborhood(imageHashMap, imageId,
                                                                layerNum, position)
                                (label, neighborhood)
                        )
                .toList  // now we have a list of (label, neighborhood) for all points on this layer
}.toDataSet.sample(12500).persist()

// take one foreground and one background point, save their neighborhood as pixture
saveAsPicture(res.filter(label == 1).take(1))
saveAsPicture(res.filter(label == 0).take(1))
```

```
// now we transform the image by mirror and rotate, and save them as json file
res.flatMap { (label, neighborhood) =>
        List((label, neighborhood), (label, mirror(neighborhood)),
        (label, rotate(90, neighborhood)), (label, mirror(rotate(90, neighborhood))),
        (label, rotate(180, neighborhood)), (label, mirror(rotate(180, neighborhood))),
        (label, rotate(270, neighborhood)), (label, mirror(rotate(270, neighborhood)))}
    .write.format(json).save("output")
```

Discussion

I use both pairRDD and DataSet:
I represent image data as hashmap(which is broadcasted), that maps (imageId, layerNum) to pixel array of this layer.
I represent distance data as pair rdd, where key = (imageId, layerNum), value = distance array of this layer.
And then

1). I process pairRDD of ((imageId, layerNum), 2D layer);
2). generate neighborhood array of each pixel on one 2D layer;
3). flatMap from layer to pixel, and
4). sample 12500 pixels
5). At last, I convert them(RDD) to DataSet, where each row is a pixel. This is because I have to save them to output file, where each row should be one pixel. Also, DataSet is faster than RDD when saving to file.

(1). I use pairRDD when processing the distance file and generating neighborhood array. After this and sampling 12500 points, I convert it to Dataset, rotate and mirror the dataset, and save the dataset as json file.
(2). When processing pairRDD, each row contains distance values of one 2D layer. After processing, generating neighborhoods array and sampling 12500 pixels, I convert them to dataset, now each row is only one pixel.
(3). No, I don't store each image separately. ImageId is in the key of pairRDD/Dataset,

Performance

100,000 records of 5 workers: 350 seconds
100,000 records of 10 workers: 288 seconds

1,000,000 records of 5 workers: 768 seconds
1,000,000 records of 10 workers: 706 seconds

Note: The 7-layer 21 x 21 pictures of foreground/background points are included in the folder "foreground_pictures" and "background_pictures".