# Final Project Report for CS 184A/284A, Fall 2019

**Project Title:** Cell Counting Method Based on Unsupervised Machine Learning and Neural Network
**Project Number:** 23

**Student Name(s)**
Kejia Qiang, 70723451, kqiang@uci.edu

## 1. Introduction and Problem Statement

The project aims to predict (count) the number of cells in a given hemocytometer microscope view, using DBSCAN to cluster cells and CNN to classify/identify them. The model will take a microscope view of hemocytometer with cells as input, then predict/count the number of cells in this view, as well as mark them in the raw image.

## 2. Related Work:

The earlier work[1] on this problem used DBSCAN for automatically object detection. In biology and bio-medicine K-Means, and its adaptive and fuzzy variants, are in widespread use, because of its robustness to low image resolution and noise in the images. But in this problem, density-based approaches are more efficient than K-Means based approaches.

DBSCAN is a popular density-based cluster method, its principal idea is to find dense areas and to expand these recursively in order to find clusters. However, since it is density-based, we need another supervised-learning approach to detect target and noise among all clusters.

CNN[2] have shown a great success in single-label image classification. Especially for limited target, CNN only needs few training data to give accurate predictions among “target” and “noise”.

---

[1] Bodenstein, Christian, et al. "Automatic object detection using dbscan for counting intoxicated flies in the florida assay." *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016.

[2] Wang, Jiang, et al. "Cnn-rnn: A unified framework for multi-label image classification." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

**3. Data Sets**

Training dataset is from [www.kaggle.com/paultimothymooney/blood-cells](www.kaggle.com/paultimothymooney/blood-cells)

It is initially used for training CNN classification model to distinguish different immunity cells. But here we used it to train our cluster model as well as CNN.

Here is one of image in the dataset. We can see that there are multiple blood cells on the image, including red cells and painted immunity cells.

Along with images, details about cells on image are given by individual xml files.

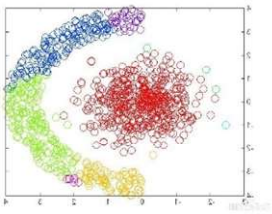
```
<object>
  <name>RBC</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>216</xmin>
    <ymin>359</ymin>
    <xmax>316</xmax>
    <ymax>464</ymax>
  </bndbox>
</object>
<object>
  <name>RBC</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>77</xmin>
    <ymin>326</ymin>
    <xmax>177</xmax>
    <ymax>431</ymax>
  </bndbox>
</object>
```

Every single cell is  marked in the file, with their bandbox, quantity, and difficulty. So, the dataset is convenient for us to train cluster and CNN model.

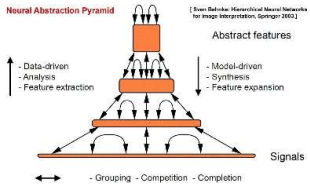
For CNN model, we cut the image into pieces along their bandbox and each piece contains only one cells. Therefore, dataset for CNN model is generated from origin dataset.

**4. Description of Technical Approach**

Density-Based Spatial Clustering of Applications with Noise: The algorithm is a kind of unsupervised machine learning method. It will cluster the dataset based on density. It is helpful when we do not know which part of dataset has relationship to other parts. The DBSCAN gives us a clear macro view of dataset. In our application, the DBSCAN will pick out all target cells, cell fragments, and other noises among environments. Input: a multi-dimension dataset. Output: labeled dataset.



Convolution Neural Network: A modified neural network, with features of Sparse Interactions, Parameter Sharing, Equivalent Representations. The CNN has advantage in processing grid-like features, like image pixels and audio. I plan to use CNN to identify each data frag if they are valid cells or noises. Therefore, a supervised learning process is required before running the model. That is, ask users to upload a sample image for cell needed to be counted to train the model, before reading users' dataset.




**WORKING FLOW**

a. Data Preprocessing (if empty grid applies, we can minus empty grid image from raw image to reduce noise)
b. Detect edges. (Enhance characteristics of image and reduce dimension)
c. Filter points. (reduce dimension again, from 2D image to 1D array)
d. DBSCAN
e. Reduce noise by removing small clusters
f. Finalize cluster
g. Sampling for CNN
h. Quickly train CNN by using few target and noise data
i. CNN detect which clusters are targets and which are noise
j. Finalize result

**4. Software**

| Stage | Part | Ownership | Purpose |
|---|---|---|---|
| Cluster | Preprocessing | myself | All data are processed by me, to fit following training |
| | Dimension Reduction | myself | I designed a filter algorithm to pick datapoints from edge magnitude map. And turn it to one-dimension array |
| | DBSCAN | Scikit-learn | I used the packed algorithm from Sklearn cluster, then adjusted parameters of algorithm. |
| | Finalizing | myself | removing small clusters to reduce noise |
| CNN | Sampling | myself | Sampling cell image slice from raw image, by clustering result |
| | CNN training | Pytorch | The CNN library is from Pytorch, and the model structure is based on LeNet-5. I modified a few parameters to improve performance. |
| | Finalizing | myself | Reform variables from previous part, and visualize the result |

Also:

Dataset external code: used to organize dataset, retrieval label information and operate in file system.

Verification code: used to test accuracy/performance of algorithm

Encapsulation code: used to organize module between python file and notebook

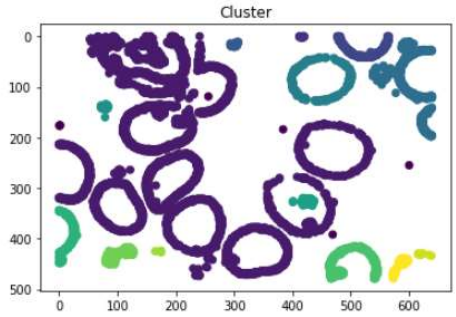
**5. Experiments and Evaluation**

For DBSCAN, there are two main parameters that decide the performance of algorithm.

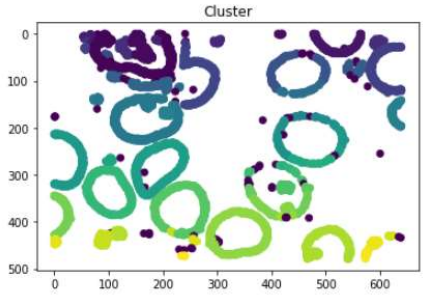
a. Epsilon (eps). The threshold of neighbor distance. Below the distance, two points will be considered as one cluster. If eps is too large, then more points will be incorrectly classified into one cluster. Respectively, if eps is too small, then one cluster might be separated into several small clusters.
b. min_sample. The minimum number of samples to be considered as a core object for cluster. If min sample is too large, then the number of clusters is not enough. If min sample is too small, then there will be redundant new clusters that belongs to others.

To approach the best parameter pair, I manually tried several combination.

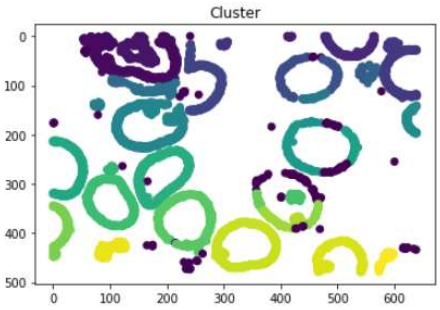
When eps = 20, many circles are classified into one big cluster.



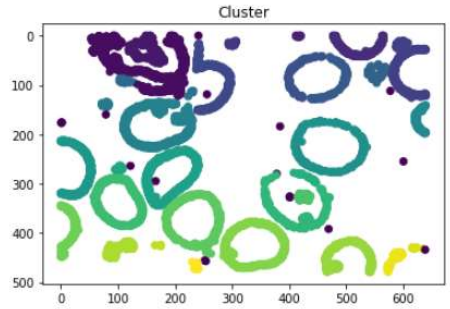

When eps = 5, redundant clusters are in the middle of circles, which make it inconsistent and separate.




When min_sample = 20,  some cluster that not strong may be divided into parts, because of lack of cluster core.




Eventually, I found when eps = 12 and min_sample = 5, the cluster looks normal



After validation on dataset, the accuracy of DBSCAN reaches 0.9.

Formula: accuracy = 1 – abs(total_cells-cells_detected)/total_cells

For CNN step, my model is based on LeNet-5, a mature and well-performed model used to classify small images into single labels. A popular example of LeNet-5 is identifying hand-writing numbers. I modified several parameters to fit the problem dataset and improve performance when do fast-training.

Layer1: convolution layer, 1 kernel, size 5x5, and 28x28 feature mapping, with 2 padding

Layer2: max pooling layer

Layer3: convolution layer, 16 kernel, size 5x5, and 14x14 feature mapping

Layer4: max pooling layer

Layer5: convolution layer, 120 kernel, size 5x5

Layer6: full connection layer, 120 nodes connected to layer5

Layer7: full connection layer, 84 nodes to 2 output, logsoftmax

Accuracy: 98% image frag successfully identified.

**6. Discussion and Conclusion**

The project goes smoothly. The DBSCAN can pick up most points of interest and CNN can identify them correctly. One insight is the difference between supervised and unsupervised learning. In the first step, given an image with no additional information, the computer does not know what it wants, which means training data or loss function help it. But a density-based cluster method can put points together. In the second step, there is a clear object to achieve. We have samples from clusters and given target. Thus, a neural network can match them well. I used two method separately to accomplish a complex process. The appropriate strategy makes whole project goes smoothly.

Furthermore, sometimes DBSCAN can not identify overlap cells correctly, that is, one cell is over another one. So, it usually underestimates the number of points of interest. We can reuse CNN to identify overlaps cells and giving prediction to number of cells in a tuple, instead of only classifying them to target and noise. The approach is clear, the CNN can be sensitive to all concave edges. The number of concave edges approximately equals to the number of cells overlaps each other.