Cryo-EMParticleCluster

A repo for cryo-EM particle clustering

Report for Cryo-EM particle cluster project

Introduction

Single-particle reconstruction in cryo-electron microscopy (cryo-EM) is a powerful technology to determine three-dimensional structures of biological macromolecular complexes in their native states. Recent advances in the detector and high-performance computing makes researchers capable of reconstruction an nearly atomic resolution 3D model. The process of reconstruction can be divided to two stages: first, enhancing the signal noise ratio(SNR) of collected cryo-EM particle images; second, reconstruct 3D model and make refinement using the SNR enhanced images. In this report, we will report an experiment finished the first part, which is the image SNR enhancing part.

This is complex problem under several all-known facts: a) a kind of proteins under physical has dynamic conformation, which means it may favor a few conformations. b) the 2D projected images is the result of projection direction and in-plane rotation. There are two popular approaches for initialize classification of 2D projection images: multi-reference alignment(MRA) and reference-free alignment(RFA). In MRA, in 2D image alignment step, the algorithm rotate and translate each image according to its reference, and that is why it is called multi-reference. The distance between each image and its reference can be measured in a way, therefore, it is easily to use these distance to cluster images like K-means. Software using this strategy is SPARX, which use a algorithm called iterative stable alignment and cluster(ISAC). Another strategy is RFA. This approach attempts to find a way applying rotations and translations on images that minimize their deviations. Another famous software in this field using this approach is SPIDER.

Some other approach are introduced to speed the cluster and increase accuracy. One is called multivariate statistical analysis(MSA), which first project the images into a smaller subspace. EMAN2 combines MSA with MRA. Another approach is the use of rotationally invariant transformation(RIT), which transformed the same invariant map if they only differ in in-plane rotation. This stragegy is quite like the thoughts in minimal representation in a cycle string. For example, two cycle string '-eat-', '-ate-' has the same minimal represent 'ate' with the minimal lexicographic order.

In this project, our strategy favors MRA, and make some modifications. As well known, K-means algorithm is a classic way in multi-reference based clustering. And the obvious properties of K-means algorithm are that it works well when K is guessed correctly and the data is separable, converge to local minimum, and clusters tend to be hyper spherical shaped. And also, K-means does not monitor the class size, which may result void class. To solve this problem, SPARX use EQK-means(equal size), which force equal size of class in every cluster. However, the projection direction and in-plane rotation may not be uniformly distributed. Therefore, we tried another modified K-means algorithm called AC(adaptively constrained)K-means, which balance the class size adaptively with classification accuracy.

Input

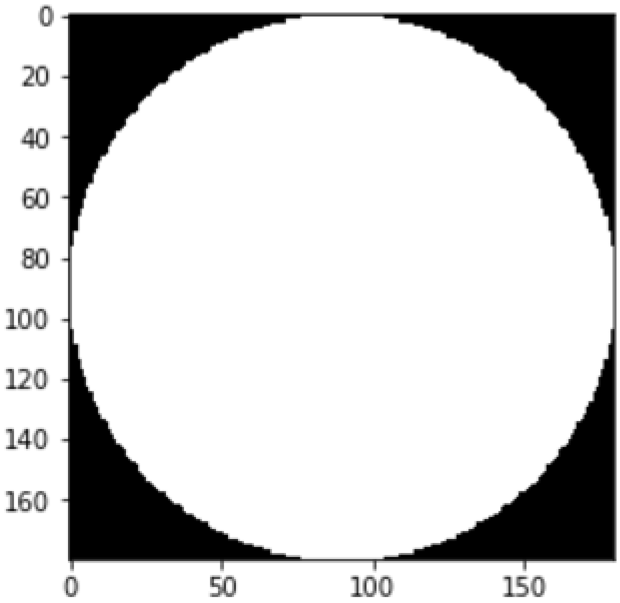
The input to our project is k(3710 X 3710) images, each with 100~200 particle images(180 X 180) in it.

Algorithm

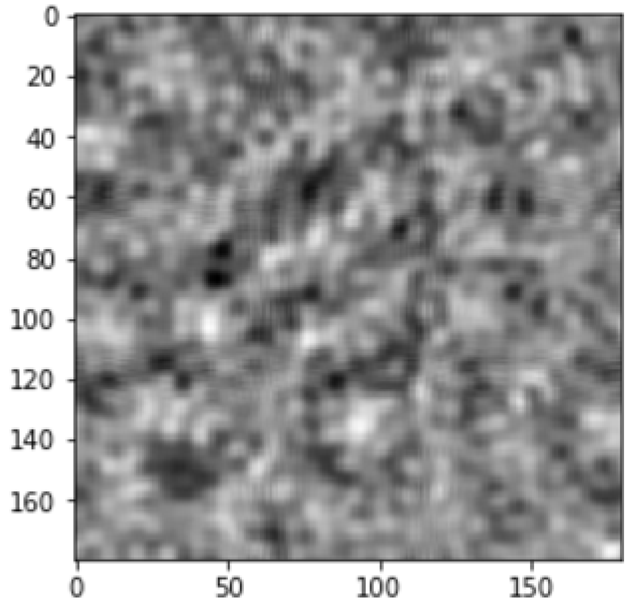
CTF correction

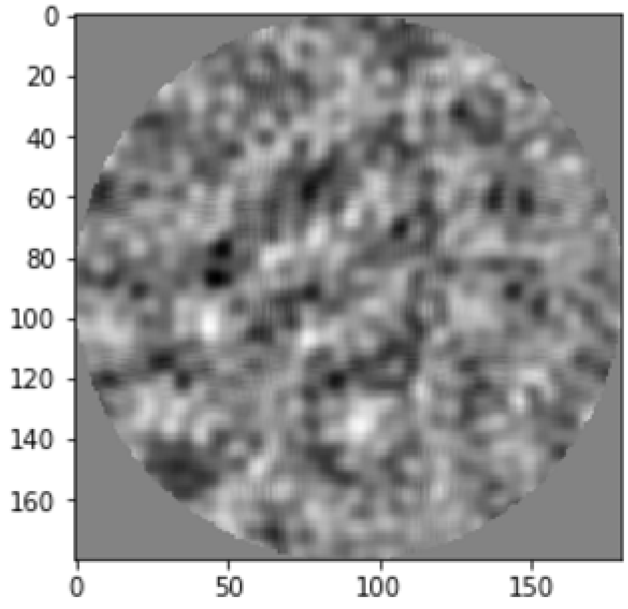
Mask(not finished fourier low pass filter):

This figure is masked by bit wise image:



An example:

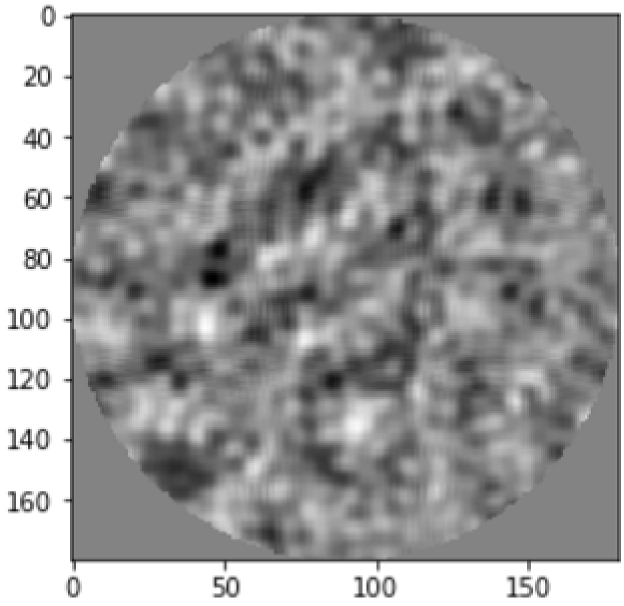


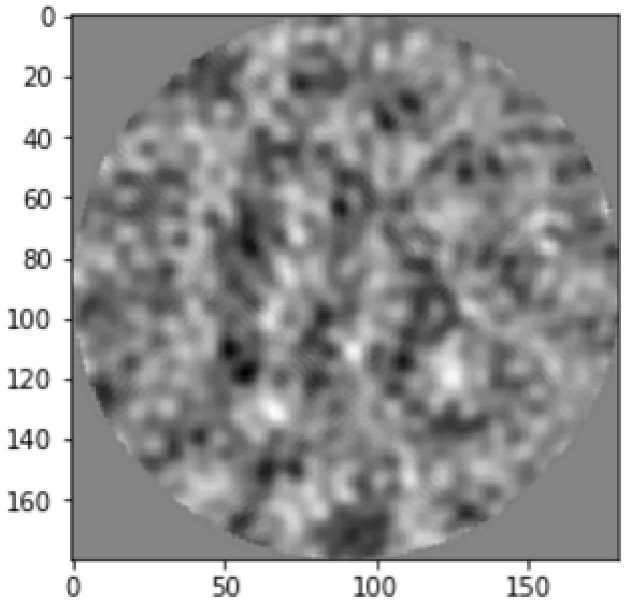


Rotation:

Rotation is based on masked image.

An example:





ACK-means

Let $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}\}$ represent a series of images to be clustered. Each class with a reference of $\mu_\mathbf{j}, for\ j = 1, 2, \ldots k$, the traditional goal of K-means is to minimize the cost function $\mathbf{J}$,

$$\mathbf{J} = \sum_{\mathbf{j=1}}^{\mathbf{k}} \sum_{\mathbf{p_i=j}} \mathbf{dissim}(\mathbf{x_i}, \mu_\mathbf{j})$$

where partition vector $\vec{\mathbf{p}} = (\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n})$.

To take the size of cluster and balance the classify accuracy, we add an adaptive constraints to the cost function:

$$\mathbf{J} = \sum_{\mathbf{j=1}}^{\mathbf{k}} \sum_{\mathbf{p_i=j}} \mathbf{dissim}(\mathbf{x_i}, \mu_\mathbf{j}) + \lambda \mathbf{ss^T}$$

and $\mathbf{s} = (\#images\ belong\ to\ 1, \#images\ belong\ to\ 2, \ldots, \#images\ belong\ to\ n)$

λ is a non-negative parameter that reflect adaptive constraint of class accuracy.

Here we give some denotes and conclusions:

$$s'_{p_i} = s_{p_i} - 1$$

$$\delta_{hj} = \begin{cases} 1 \;\; if\, h = j \\ 0 \;\; else \end{cases}$$

$$\begin{aligned} \mathbf{p_i} = & \quad arg\,\mathbf{\min_j}\{\mathbf{dissim(x_i}, \mu_\mathbf{j}) + \lambda \sum_{\mathbf{h=1}}^{\mathbf{k}} (\mathbf{s'_h} + \delta_\mathbf{hj})^\mathbf{2}\} \\ = & \quad arg\,\mathbf{\min_j}\{\mathbf{dissim(x_i}, \mu_\mathbf{j}) + \mathbf{2}\lambda\mathbf{s'_j} + \lambda + \lambda \sum_{\mathbf{h=1}}^{\mathbf{k}} \mathbf{s'^2_h}\} \\ = & \quad arg\,\mathbf{\min_j}\{\mathbf{dissim(x_i}, \mu_\mathbf{j}) + \mathbf{2}\lambda\mathbf{s'_j}\} \end{aligned}$$

and 2λ was rewrite as :

$$2\lambda = \beta \frac{d_c}{\lfloor n/k \rfloor}$$

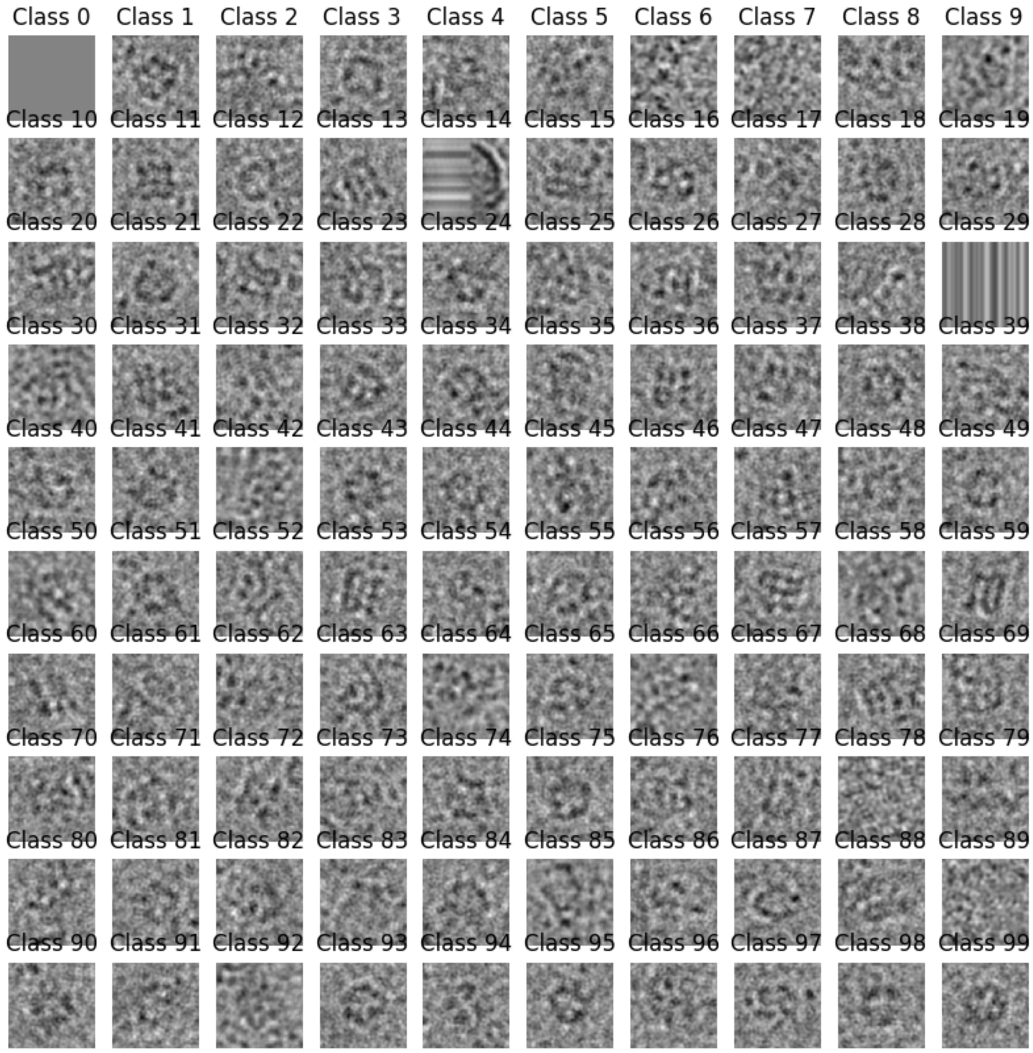
where d_c describes the change of pixel intensity scaling, whereas the second parameter β decides the weight on the class size whose value is independent of data scaling. The constant $\lfloor n/k \rfloor$ is the class size if all images are partitioned equally. β is set by experience as 0.5.

Pseudo-code of ACK-means

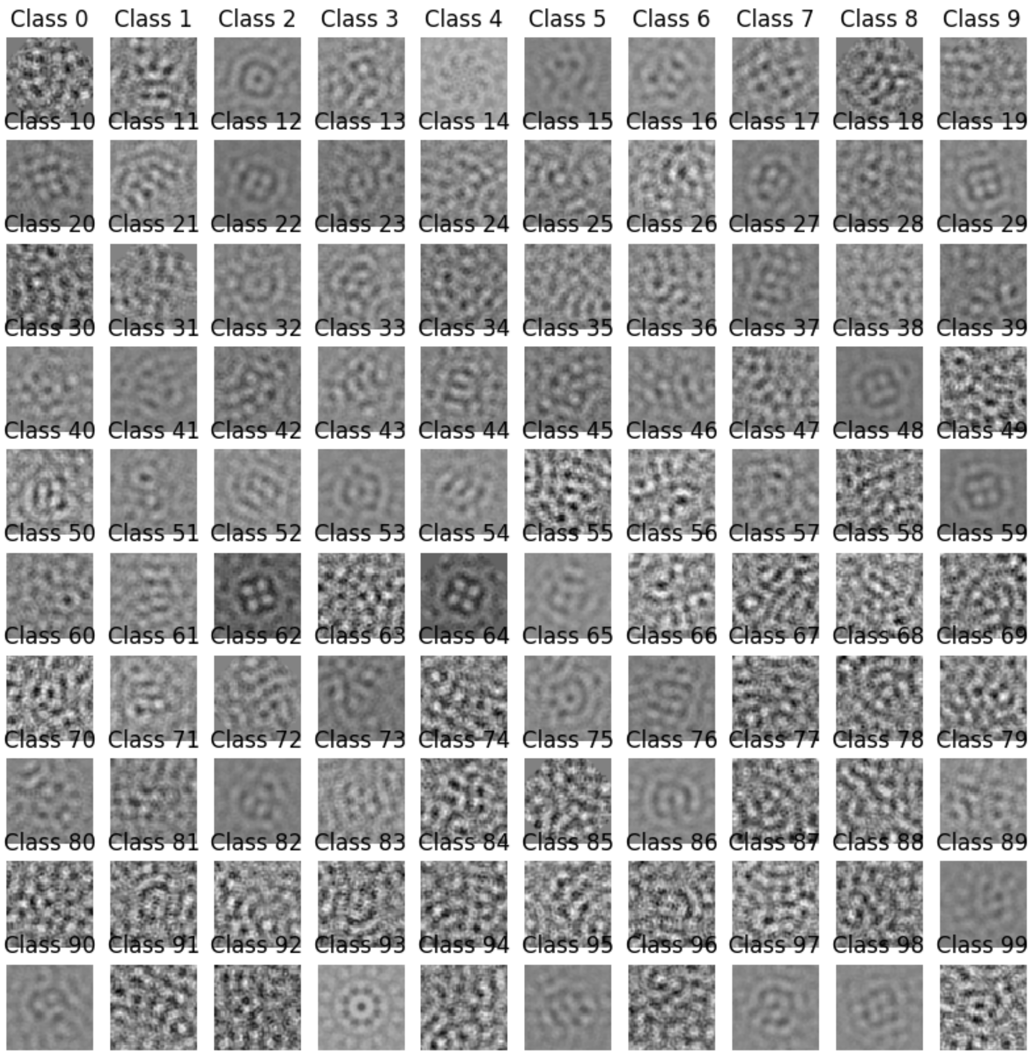
```
#ACKMEANS#

sigma_zero: minimum fraction of data change criteria
beta: the weight on the adaptive constraint term
k: number of class

# Initial assignment
    Update class
        for each datapoint
            find closest centroids index i
            compare new class to previous ones
                if not equal
                    flag = true
            assign/update this number to its class
    Update centroids
        for each centroids
            cent = average(all class member)
# Update cycle
while sigma > sigma_0

    calculate the change of pixel intensity d_c
    by randome select images
    t_m = max dist(image_m, centroid) -
                         min dist(image_m, centroid)
    d_c = average(t_m)
    lambda = d_c * beta / (2 * [n/k])
    class_Assignment_old = class_Assignment

    for each datapoint
        s_prime = # other datapoint whose class is the same as
                    this one
        update classAssignment

    update centroid
        for each centroids
            cent = average(all class member)
```

Results

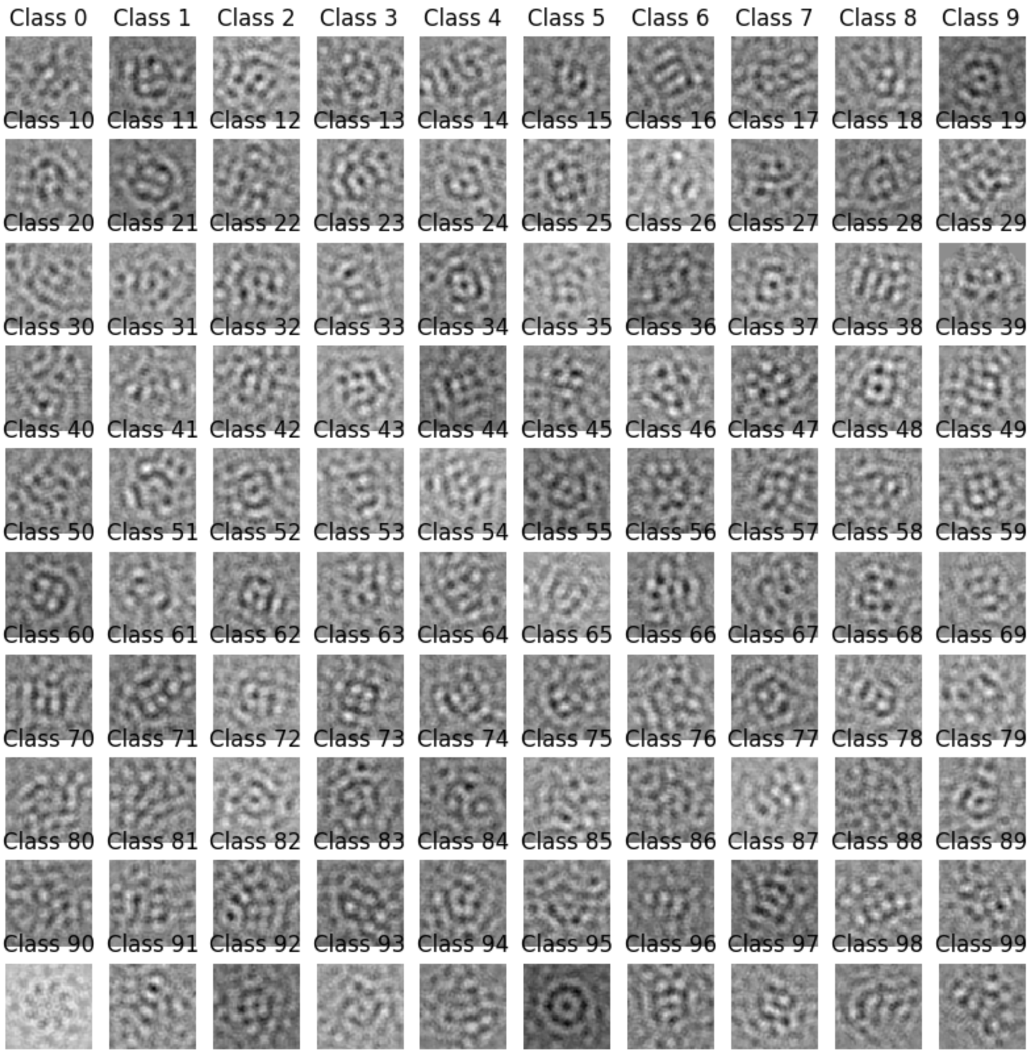
Simple K-means(CTF processed):



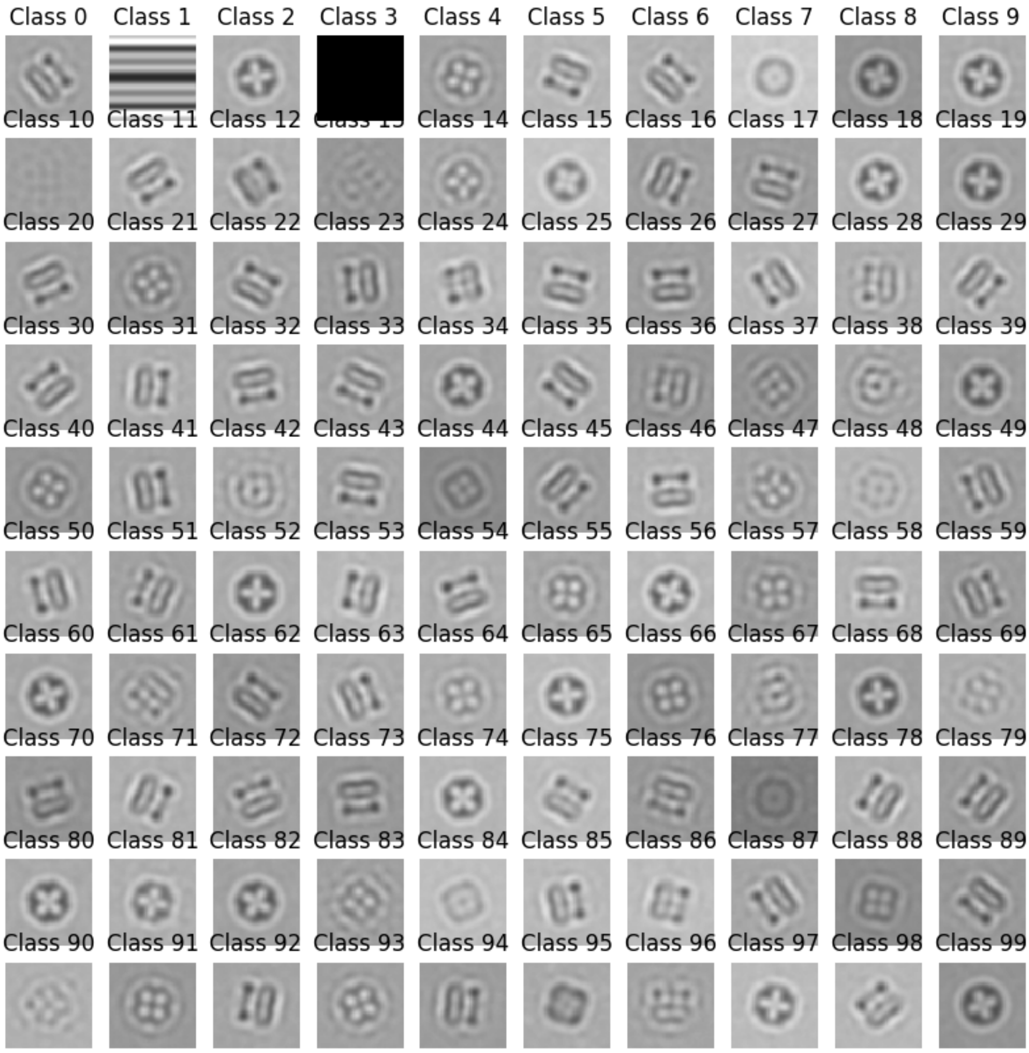
Rotation amplified result(CTF processed)



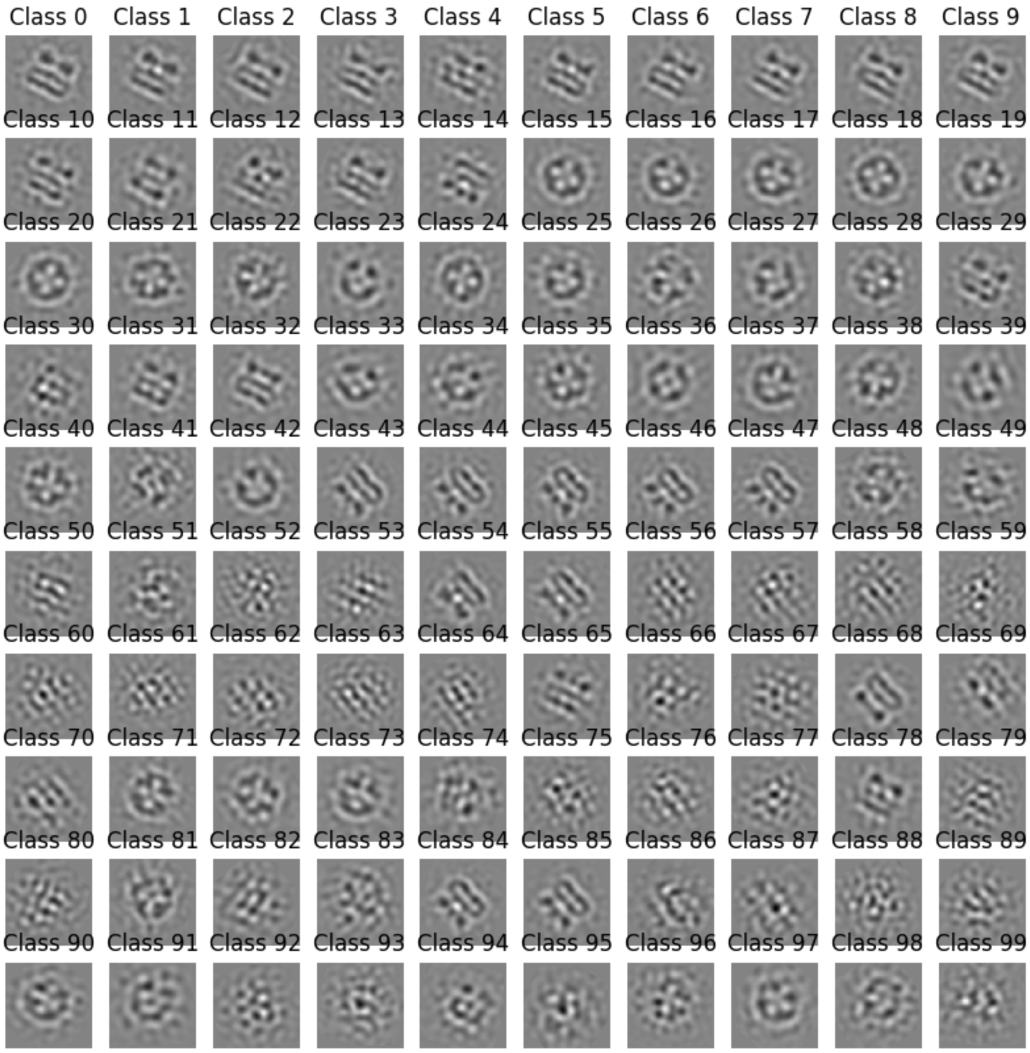
Rotation amplified ACKmeans result (CTF processed, only using 1000 input samples, while rotated using ~ 60000)



Rotation amplified result(CTF processed)



EMAN2 cluster result(CTF processed, MSA + MRA)



Discussion