Communications in Information and Systems
Volume 19, Number 3, 321–341, 2019

An automatic particle picking method based on Generative Adversarial Network

Fang Kong, Xirong Li, Qing Liu, Chuangye Yan, and Xinqi Gong

Cryo-electron microscopy (cryo-EM) technology has greatly facilitated the development of biology and medicine. Particle picking is a critical step in the processing of cryo-EM micrographs. However, achieving fast particle picking remains a bottleneck because the micrograph has a very low signal-to-noise ratio (below 0.1), large image size (usually 4k $\times$ 4k), small particle sizes and large numbers of particles. In this paper we propose a cGAN-based approach to mark out particle regions. We propose a data synthesis method to generate training samples thus there is no need to prepare particle samples from original micrographs. This data synthesis method will be very helpful when applying on different kinds of particle micrographs. We use the mean squared loss to improve the cGAN effect. In order to better demonstrate the performance of our method, we tested on the public dataset EMPIAR. The results show that our method can achieve fast and accurate automatic particle picking, and the performance is better than other known methods.

1. Introduction

Cryo-electron microscopy (cryo-EM) technology along with single particle analysis is the state-of-the-art in structural biology [1] for achieving near-atomic resolution 3D reconstructions [3, 4, 2] with un-crystal biomolecule samples. Achieving high-resolution structures requires a large number of biomolecules (also called particles) projections taken from micrographs.

The main steps of cryo-EM micrograph processing can be roughly divided into six steps, including CTF correction [5], particle picking, 2D clustering and class averaging, initial structure calculation, 3D clustering, structure refinement and resolution determination [1]. Particle picking is a key step to obtain plenty of orientations of the particles, which is critical for the 3D construction.

However, particle picking is a hard problem. In terms of the quality of micrographs, to protect particles, low electron exposure is applied in experiments which results in low signal-to-noise ratio (SNR). The SNR of a micrograph is lower than 0.1. Only noise information can be seen from the histogram. From the perspective of electronic signal imaging, the high-frequency signal reflects the details of the particle structure. In order to recover high 3D resolution and sophisticated structure of biomolecules, low defocus is applied to retain as much high frequency information as possible, which causes the low frequency information to be lost more. The micrographs acquired are with low contrast [1]. That means these micrographs are less recognizable to the human eye. In terms of data volume, a micrograph is often with a high resolution of 4k $\times$ 4k which means there may be hundreds of particles on a micrograph. More effective particle picking methods are required to relieve the bottleneck of current methods.

Particle picking method has been widely studied in biology and computer science. There are two main kinds of method: semi-automatic method and fully automatic method.

Semi-automatic Particle Picking is based on template matching [6]. This method needs to provide a small number of samples that have been selected as templates. Those templates are manually picked from micrographs or projections made from known 3D structures. For all candidate positions as input, the output are regions with high cross correlation with templates. Template matching based methods [9, 4, 7, 8] have been widely used in most existing cryo-EM micrograph process software including EMAN [10], RELION [11] and cryoSPARC [12]. These methods are often time consuming.

Till now, fully automatic particle picking approach can be divided into three categories.

The first kind of automatic particle picking enhances image information, which makes it easier to distinguish the particle-like regions from non-particle regions.

In the reference [13] a binary segmentation approach was proposed for particle picking. Image enhancement and thresholding were applied to get a binarized map. Then morphological segmentation methods are easy to apply on these maps to get particle locations. The autocryopicker [14] is recently proposed based on image enhancement, which contains a series of methods for image denoising, image enhancement, clustering, and morphology operations.

These approaches use traditional image processing methods such as denoising and contrast enhancement followed by particle localization methods. However, these methods can only be used on regular particle shapes such as circulars and rectangles [14].

The second kind of automatic particle picking regards particle picking as a classification problem. The common approach is to train a classifier on a labeled dataset, then sliding a window on the image to obtain a map where each pixel value represents for whether it is a particle pixel.

APPLE picker [15] used a support vector machine (SVM) [16] as the classifier to obtain a binarized score map. The SVM is trained each time when performing on a new particle dataset, which means that the parameter adjustment of SVM is required to get good results on a new dataset. The Apple picker takes 11 seconds to process each micrograph on GPU according to their article, which is not fast enough. The implementation on the CPU takes 3 hours to complete the processing of 84 micrographs.

Convolutional Neural Network (CNN) [17] based methods include Deep-Picker [18], DeepEM [19] and [20]. These methods need a lot of particle samples for the network training. For different shaped particles, suitable particle datasets are needed to train the network [21].

The third kind of automatic particle picking uses regression method, which can directly obtain a score map from the input image and no need for sliding window. crYOLO [22] utilizes the first generation YOLO [23] network to detect particles. crYOLO performed particle picking very fast and attained better results compared to the original YOLO model when working with small objects as particles in those test datasets. But its regression and classification approach will cause high localization error [24]. A Deep regression based method [21] introduced a fully convolutional regression network (FCRN) to predict the probability map of particle centers and then classify those candidates to pick particles. Only results on EMPIAR-10017 dataset are shown, and the time cost and ability on other datasets are not mentioned.

Table 1: Comparison of method limits

Method	shape limited	speed	particles for training
semi-automatic method	yes	slow	needed
the first kind of automatic method	yes	slow	no need
the second kind of automatic method	no	fast	needed
the third kind of automatic method	no	fast	needed
our method	no	fast	no need

In this paper, we propose a particle picking method which is fast and does not need real particle samples for training or as templates. Our work belongs to the image enhancement solution in fully automatic category. We propose a pipeline where a Conditional Generative Adversarial Network (cGAN) [25] is used to mark out particle regions. The cGAN model generate marks on

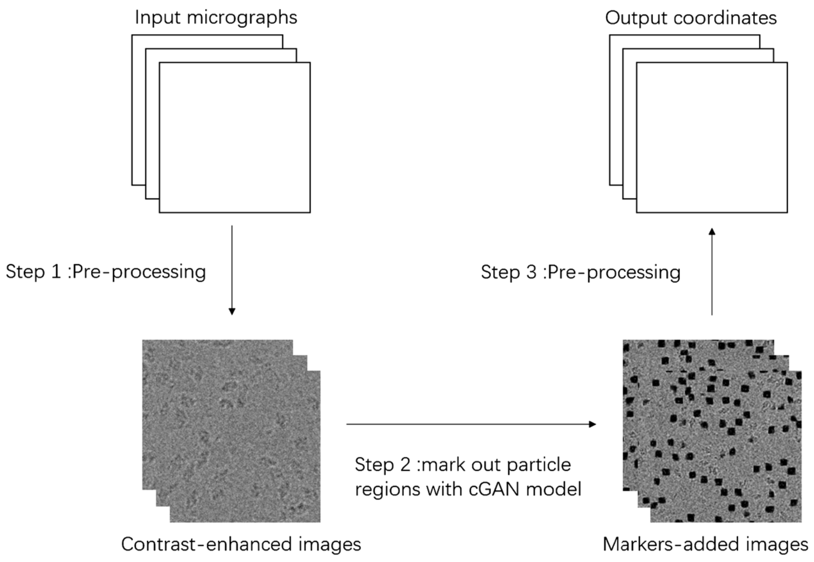


Figure 1: **Overall workflow.** The inputs are original micrographs with ".mrc" suffix. The first step, "Pre-processing", converts these files to "png" format and uses histogram equalization for contrast enhancement. The second step, "Mark out particle regions using the cGAN Model," will add black markers on these png-format images to indicate locations of particles. The final step uses several algorithms to recover coordinates from these markers-added images.

particle regions. The pixel value of particle regions will be decreased. The image can be easily divided into particle regions and non-particle regions according to pixel values. We are the first to introduce data synthetic method thus particle samples are not required for training the network. The advantage of using the synthetic data method is that even if the pre-trained model is not applicable on a new type of particle, it is convenient to give a sample that is roughly equivalent in number and size to the new particle, regardless of the particle shape. Our method learned from synthetic images and achieves good experiment results on different raw micrograph datasets.

2. Method

Our pipeline contains three main steps: pre-processing, mark out particle regions with cGAN model and particle localization. The overall workflow is shown as Figure 1. We will introduce these steps in each section.

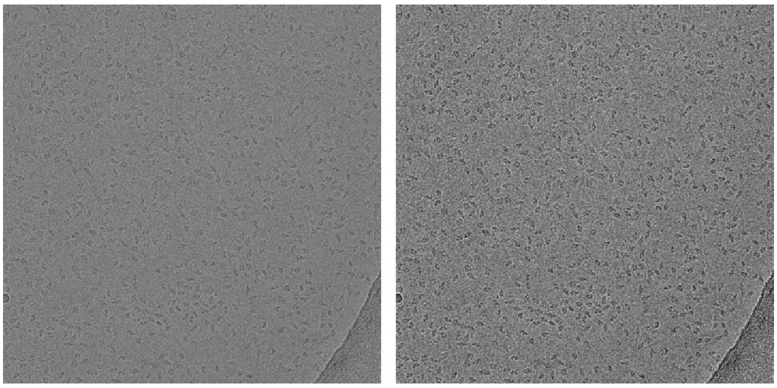

(a) The input image (b) The image after pre-processing

Figure 2: **One original image and the corresponding one after pre-processing.** The contrast of the input micrograph image will be significantly improved after pre-processing.

2.1. Step 1: pre-processing

A standard cryo-EM image is stored in the Mixed Raster Content (MRC) format. We converted cryo-EM images in the MRC format into PNG format using EMAN2 [25]. And a contrast adjustment step was used before the PNG format image input into the GAN.

In the contrast adjustment step we used histogram equalization method. Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of images, especially when the data is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. In our case, the contrast of the original micrograph is very low, and the pixel values of particle and non-particle regions are close, making these regions difficult to distinguish. Using the histogram equalization method will help improve the contrast of these micrographs. In practice, the intensity values of the micrographs are adjusted to new values in a condensed smaller range by using the *adapthist* function provided by opencv, which can also be found in *cv2* python library. Figure 2 shows one original image and the corresponding image after pre-processing.

2.2. Step 2: mark out particle regions with cGAN model

In order to increase the speed of particle picking, we consider not judging each candidate region in the micrograph, but marking out particle regions

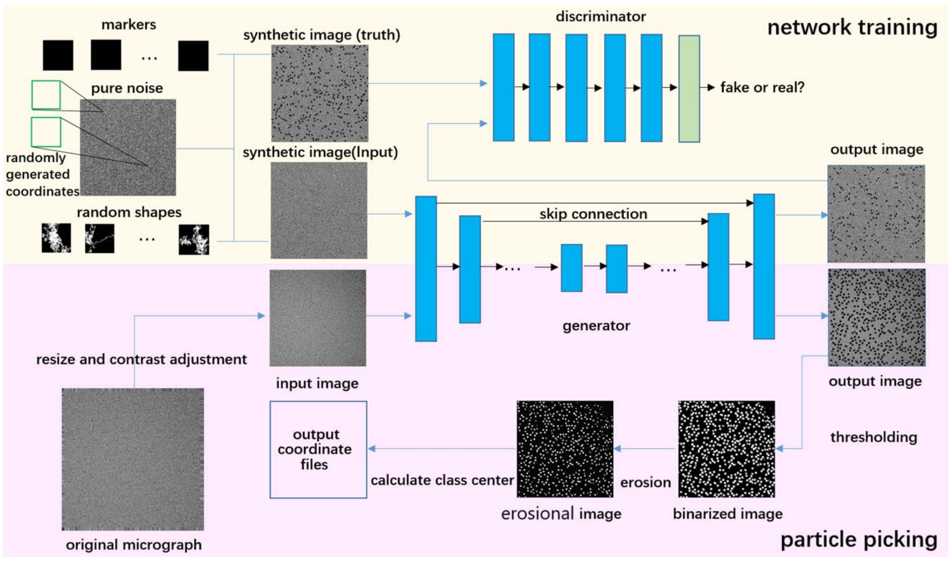


Figure 3: **Network training and particle picking.** In network training step we use pure noise images as background. For each noise image, black markers and random shapes are added sharing the same randomly generated coordinates. And thus synthesis the input and ground truth images for the network training. In particle picking phase, original micrograph is resized and contrast adjusted and then is input into the trained generator, the output image will by divided to a binarized image through a threshold. An erosion operation will be applied on the binarized image to cut off regions that are too close. Finally we obtain a coordinate file by written out class centers in the erosional image.

using cGAN network. And it's easy to restore particle coordinates directly. We improved a cGAN network based on pix2pix [26] and introduced mean squared loss to achieve our design goals. We devised a method of data synthesis so that it did not rely on real particle images during the network training phase.

2.2.1. cGAN model Our method is based on cGAN, which consists of two parts: generator G and discriminator D. The generator attempts to generate an image from random latent noise z and conditional information y. The discriminator determines whether the input image is a real image or a fake sample (generated by G). The two parts of the network compete with each other. The generator attempts to confuse the discriminator. The discriminator learns to correctly judge the authenticity of the image. The loss function is a minmax problem.

$$\begin{aligned} L_{cGAN}(G, D) = E_{(x,y)}[logD(x, y)] \\ +E_{(y,z)}[log(1 - D(y, G(z, y)))] \end{aligned} \tag{1}$$

The generator and discriminator networks are based on pix2pix [26] model.

The generator is based on u-net [27] structure, in which the encoding network and the decoding network are both 9-layer convolutional networks using the conv-batchnorm-relu approach. Additional connections are added between the corresponding layers of each encoding network and decoding network. That is, for the ith layer $(1, 2, 3 \ldots, 8)$, a connection to the $(19-i)^{th}$ layer is additionally added. The output of the i^{th} layer will not only be sent to the next layer but also be concatenated with the input of the $(19 - i)^{th}$ layer. For our particle picking task, this added connection can share position information between the input and the output.

The discriminator is a CNN containing 5 layers. The input to the discriminator is a set of images, including real markers-added images or the generated images, the output of the network is a value to each image indicating the probability that it is true or false.

As explained in [28] that the mean squared distance is better than the cross entropy loss function in the image generation task. We design a loss function in the form of a mean squared distance. For the generator, our design goal is to make the score $D(G(z))$ and the score $D(x)$ closer, where $D(G(z))$ represents the score for the generated image, and $D(x)$ is the score for the real image. For the discriminator, our design goal is to make the score of the actual image and the score of the generated image close to our preset two values a, b. A tanh function is used in the output layer. This will map the judgment given to each image to $(-1, 1)$. When we use mean squared loss as the loss function, the optimization direction for the negative sample is to make it close to 0 instead of -1. The loss function defined by the mean squared distance can be expressed as:

$$\begin{aligned} \min_D V_{MS}(D) = E_{(x,y)}[(D(x, y) - a)^2] \\ +E_{(y,z)}[(D(G(z, y)) - b)^2] \end{aligned} \tag{2}$$

$$\min_G V_{MS}(G) = E_{(x,y)}[(D(x, y) - D(G(z, y)))^2] \tag{3}$$

In addition, we introduce the L1 distance in the loss function of the generator. This method not only does not affect the main design goal of the discriminator, but also makes the image generated by Generator closer to the real image y (not just the score). This L1 loss is also called data loss. The L1 loss which is also called data loss has been found to be beneficial when mixed with GAN loss function as mentioned in [26].

Previous approaches have found it beneficial to mix the GAN objective with a more traditional loss, such as L2 distance. While L1 or L2 loss encourages the generator to produce a rough outline of the predicted object, it often fails to capture any high frequency detail. This stems from the fact that the L2 (or L1) loss often prefer a blurry solution, over highly accurate textures. We also explore this option, using L1 distance rather than L2 as L1 encourages less blurring.

$$L_{data} = E_{(y,z)}||y - G(z,y)|| \tag{4}$$

We define a, b as 1, 0. Finally, the objective function of our network can be expressed as:

$$\begin{aligned}\min_D V_{MS}(D) = E_{(x,y)}[(D(x,y) - 1)^2] \\ +E_{(y,z)}[(D(G(z,y))^2]\end{aligned} \tag{5}$$

$$\begin{aligned}\min_G V_{MS}(G) = \omega_1 * E_{(x,y)}[(D(x,y) - D(G(z,y)))^2] \\ +\omega_2 * L_{data}(G)\end{aligned} \tag{6}$$

2.2.2. Training We designed an automatic method to generate training sets. The train set contains 100 synthetic input images with a size of 512∗512 and 100 synthetic truth images with the same size. Each synthetic input image has a corresponding synthetic truth image. The synthetic truth image is a synthetic input image with black markers added for each particle position. The specific method is as follows:

We select areas that do not contain particles from some micrographs and resize these areas as background.

These background images are used for both synthetic input images and synthetic truth images. Coordinates are then randomly generated within the background image size. Images containing particles and images with added markers will share the same coordinates as particle centers. Markers with specific sizes are black squares whose pixel values are zero. These random patterns are generated by random walk algorithm (Algorithm 1).

In our experiments, the size and the number of random shapes are random values within a certain range. The product of the size and number of

Algorithm 1: random walk algorithm

```
Input: max step: n
Output: random shape: img
Initialization
Identify initial coordinates (x,y) to be (0,0), pix[ ] to be
zeros(n,2),neighbour[ ] to be [-1 -1;-1 0;-1 1;0 -1;0 1;1 -1;1 0;1 1]
/* the array pix[ ] is used to save all walk steps, the coordinates of each
step will be recorded
the array neighbour represents for all possible movement with one step */
for $i = 1; i <= n; i = i + 1$ do
    rand_num ← rand(1,8) /* r is a random integer from 1 to 8*/
    x ← x + neighbour(rand_num,1)
    y ← y + neighbour(rand_num,1)
    pix[ ]← (x,y) /* append the coordinates of this step*/
end
pix[:,1] ← pix[:,1] - min (pix[:,1]) + 1
pix[:,2] ← pix[:,2] - min (pix[:,2]) + 1
/* Make image index values start from (1,1) */
maxy ← max(pix[:,1])
maxx ← max(pix[:,2])
img ← new_image (maxx.maxy)
/* Create a new image according to the max coordinates, the value of each
pixel is 0 by default*/
for each item ∈ pix[ ] do
    /* item is a turple of (x,y)*/
    img(item) ← 255
end
```

generated random graphics is not larger than the image size. In order to ensure the random graphics do not overlap, graphics that are too close to other existing random graphics centers will not be generated.

When generating an image containing particles, the noise image will minus one random shape at each coordinate. This method simulates the case where the intensity value of the particle region is lower than in other regions. The generation method (Algorithm 2) refers to the theoretical results of [4, 29]. When generating an image with added markers, markers and noise image will perform a logical operation "and" at the same coordinate. Pixel values of these marker added regions are zero. All random values are generated in a specific range in the experiment, and we found that it is very effective to select a suitable random range for the test data.

Algorithm 2: data synthesis algorithm

```
Input: random shape images: R_I[ ], noise image: N_I
Output: synthetic image: S_I
Initialization, S_I ← N_I /* do not change original noise image*/
Identify shape_diameter, shape_num to be a random value,
/* shape_num * shape_diameter * shape_diameter should be smaller than
the image size*/
image_size to be the size of N_I
for $i = 1; i <= shape\_num; i = i + 1$ do
    rand_coord ← random tuple within image_size
    rand_shape ← random item in R_I,
    resized as[ shape_diameter,shape_diameter]
    rand_fade_value ← random float value between 0 and 1
    if not overlapped then
        S_I ← S_I - rand_shape * rand_fade
    end
end
```

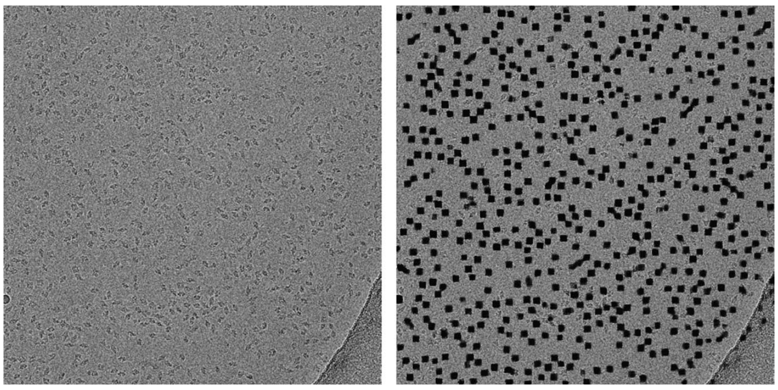
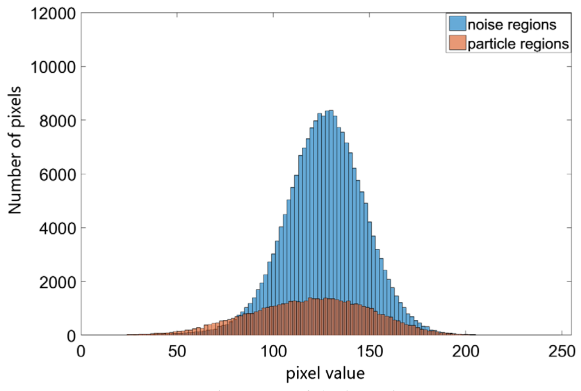
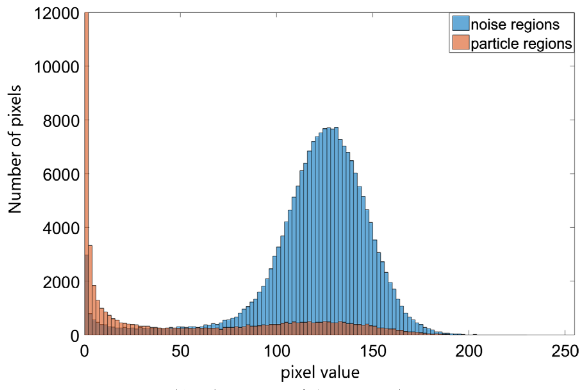

(a) Input image (b) The output image of GAN

Figure 4: **Results of one input image.** Markers are added on particle regions according to the input image. And the shape of those dark regions are close to square. This is caused by the training process.

2.2.3. Using GAN on cryo-em images Images after pre-processing will be input into the generator part of GAN model, the output are images with markers added on particle-like regions based on input images. For the design of generator is $input - gen(input)$, particle regions will be darker on the image (as shown in Figure 4). So we can get $gen(input)$ part using



(a) Histogram of the input image



(b) Histogram of the output image

Figure 5: **The histogram of input image and output image.** The horizontal axis represents the pixel value, and the vertical axis represents the number of pixels. (a) is the histogram of an input image. The particle regions and the noise regions are difficult to distinguish. (b) is the histogram of the output image. Particle regions and noise regions are easy to distinguish. The pixel values of the particle regions are mainly concentrated between 0 and 20. A large number of particle pixel values are 0.

operation $input - output$. Thus we get particle regions. Figure 5 shows how the histogram changed after markers are added on input image.

2.3. Step 3: particle localization

Images we get from step 2 will be convert into a binarized image using a threshold (by default 0.1). Pixel values of marker regions are converted to 1, and the rest are converted to 0.

Algorithm 3: particle localization algorithm

```
Input: ori image: $I$, output image from cGAN: $O$
Output: particle locations: $coords[]$
Initialization
Identify the $threshold1$ used in binarization to be 0.1, the convolution
$kernel$ used in erosion operation to be a $square$ slem with size of 5 x 5, the
average area AA to be 0, the $threshold2$ used in particle localization to be 0.5
$IO \leftarrow I - O$
for $each\ pixel\ value\ r \in IO$ do
    if $r$ is lower than 255*threshold then
        r ← 0
    else
        r ← 255
    end
end
EB ← $IO \ominus kernel$
/* $\ominus$ operator represents for erosion operation on image IO using kernel
$kernel$ */
L ← skimage.measure.label(EB)
R ← skimage.measure.regionprops(L)
Total_num←R.length
for $each\ r \in R$ do
    AA ← AA+r.area/Total_num
end
for $each\ r \in R$ do
    if r.area between [(1-threshold)*AA,(1+threshold)*AA] then
        coords[ ] ← r.centor / *append this region center position into
        coords[ ]*/
    end
end
```

A morphology erosion operation is performed on the binarized image. This operation will remove tiny regions on the image and separate regions as much as possible. Regions too small are often noise or experimental pollution. If particles are too close, their markers are likely to be connected together, so it is necessary to separate regions. After that, we use an algorithm in python 'skimage' library to label connected regions. All connected regions are assigned the same integer value on the image. We used a two-step algorithm to calculate particle positions on this labeled image. The first step aims to determine the average area of all connected regions, the second step we output all possible region center coordinates whose area are within the

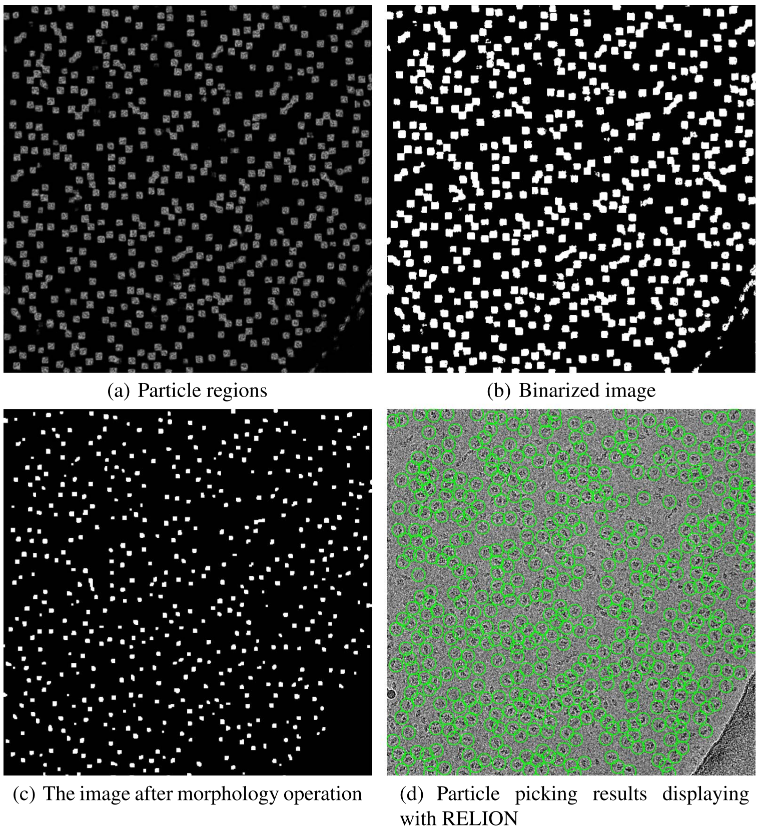

(a) Particle regions
(b) Binarized image
(c) The image after morphology operation
(d) Particle picking results displaying with RELION

Figure 6: **Particle localization results.** (a) The Particle regions image is the result of subtracting the output image by the cGAN model from the input image. Black regions in this image represents for non-particle regions. (b) The binarized image is the result of threshold binarization. White regions represent for particle regions in the image. (c) This image is obtained after morphology erosion operation on (b). (d) We use RELION software to show out particle picking results. Particles are highlighted with green circles according to the input coordinate file.

neighborhood of the average area. These steps are shown in Algorithm 3. Figure 6 shows the image obtained after each step of processing.

3. Results

We tested our particle picking method on several datasets in EMPIAR (the Electron Microscopy Public Image Archive) [30]. Note that EMPIAR is a public resource for raw electron microscopy images. Raw micrographs are provided while coordinates are usually not provided. We show particle picking results of T20S proteasome dataset directly on images. For statistical results, we use a β-Galactosidase dataset where coordinates are provided in the dataset.

3.1. Implementation

We trained our network on a nvidia-1080ti 11G GPU and we implemented the process from inputting micrographs to outputting coordinates files with a 16GB 2.2GHz Intel Core i7 CPU.

3.2. Evaluation criteria

We mainly evaluate our methods from the perspectives of Precision, Recall, F1-score and time cost. The formula of calculating Precision, Recall and F1-score are as follows:

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

$$F1\text{-}score = \frac{Precision * Recall}{Precision + Recall} \tag{9}$$

When measuring whether results are consistent with the true position, we believe that it is accurate at this position as long as it is within 1/4 of the diameter from the true center position. This judgment condition is derived from the experience provided by the biologist in actual use, that is, as long as the center position deviation is within 1/4 diameter, it will not affect the operation behind.

3.3. Experiments

3.3.1. Particle picking evaluation on the β-Galactosidase dataset

The β-Galactosidase dataset is publicly available which can be found in EMPIAR as EMPIAR-10017 [31]. This dataset contains 84 micrographs (.mrc files) and particle coordinates (.star files). The resolution of these images is 4096 $\times$ 4096, and the outermost 100-pixel area does not contain any information. We remove meaningless areas while processing. We evaluate and compare the performance of our method to other automatic particle picking

Table 2: **Comparison of particle picking results for different losses on the β-Galactosidase dataset.** Results are better when using mean squared loss

Model	Precision	Recall	F1
APPLEPicker [13]	0.73	0.37	0.49
FCRN [21]	0.79	0.71	**0.75**
Faster-RCNN [20]	0.70	**0.78**	0.74
This paper:			
GAN + *Crossentropyloss*	0.73	0.60	0.66
GAN + *meansquaredloss*	**0.81**	0.63	0.71

methods. We use synthetic data to train our cGAN model. Totally 42495 particles are picked using semi-automatic method RELION [11] software. In contrast, our method picked 32557 particles. We compared with several automated methods. The DeepPicker [18] method did not obtain meaningful results when apply on the β-Galactosidase dataset using the default pre-trained model. This problem was also mentioned in [21]. The autocryopicker method was not suitable for this data because this method was designed to detect and pick regular particle shapes [14].

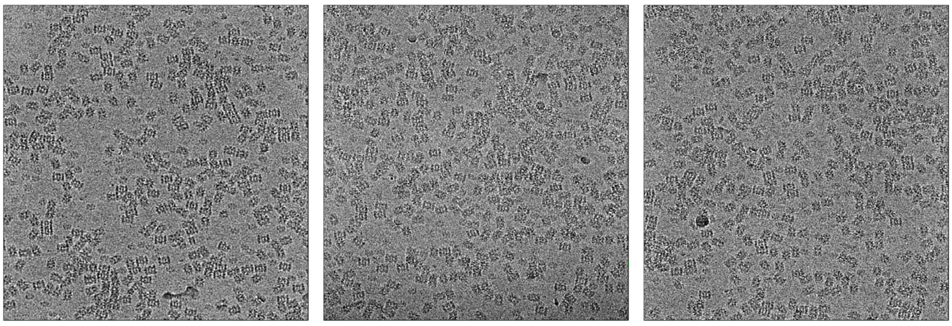
When comparing with APPLEPicker [15], FCRN [21] and Faster-RCNN [20] method, our method reached the highest precision and the F1 score was close to the best results in [21].

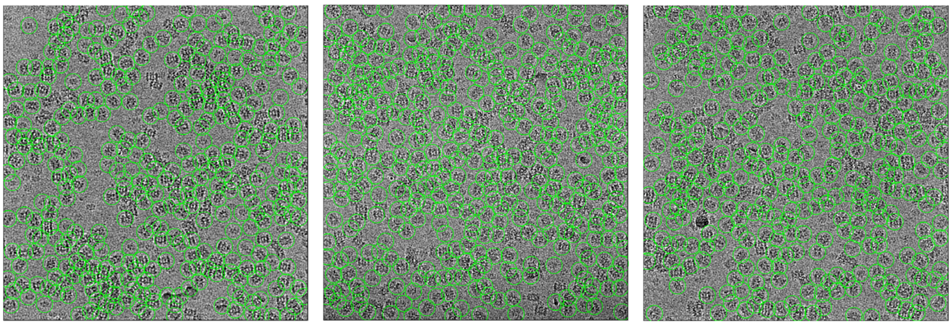
3.3.2. Robustness comparison of particle picking on the T20S proteasome dataset We compare results of DeepPicker [18] and our method when picking on the T20S proteasome dataset. The T20S proteasome dataset is publicly available which can be found in EMPIAR as EMPIAR-10188 [32]. The resolution of these images is 3710 $\times$ 3838. For this dataset do not provide reference coordinates and lack of comparison, we show several results of them.

We find that our model trained with synthetic data can also achieve better results directly on T20S proteasome dataset. The DeepPicker method did not obtain good results using default pre-trained model again. We show some representative results in Figure 7. Our method obtained good results using the same model trained on synthetic data.

4. Discussion

Looking at the results of output images, we found differences in image results. The best image can achieve a precision of 0.91 with a recall of 0.70.


(a) Input micrographs


(b) Particle picking results of our method

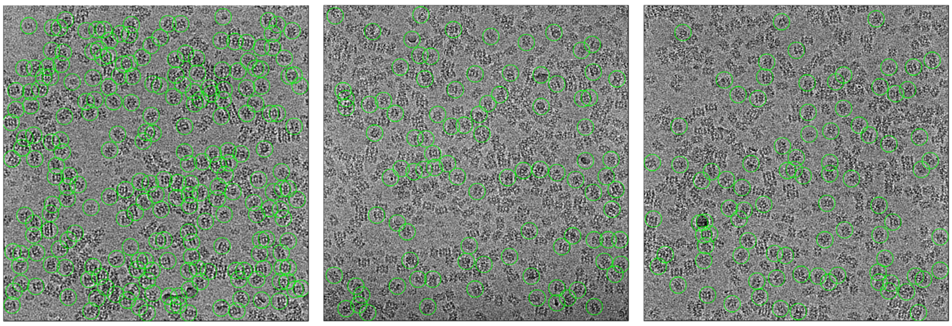

(c) Particle picking results with DeepPicker[18] using a default threshold

Figure 7: **Several results of particle picking on the T20S proteasome dataset.** (a) Three different input micrographs are shown. (b) We use RELION software to show out particle picking results. Particles are highlighted with green circles according to the input coordinate file. (c) These images are displyed in RELION software using coordinate produced by DeepPicker method. Our method achieved good results when using pre-trained model. However the DeepPicker method did not.

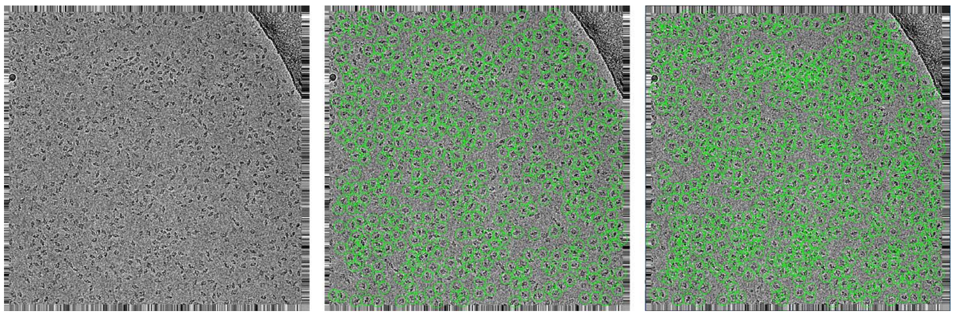


Figure 8: **Good particle picking results on EMPIAR-10017 dataset (with a precision of 0.91 and a recall of 0.70).** (a) is original image and (b) is particles circled according to our output coordinates. (c) represents for results provided by dataset. Coordinates given by dataset are regarded as ground truth. Particles are highlighted with green circles according to the input coordinate file.

However, some images have an precision of only 0.22. These images are shown in Figure 8 and Figure 9. Besides, we found that coordinates provided by dataset are highly overlapped, but in our post processing we remove all candidates that are too close to each other. These reasons lead to a relatively low average result. Furthermore, our method picked out some regions that are possible protein particles, but not mentioned in the original dataset. This results in lower precision, however some of those regions may be real particles. That is to say, our method may find some new regions which are not regarded as particles before.

Our method may have some potential drawbacks since our cGAN model marked out all regions with lower pixel values. There may exist non-particle regions with low pixel value. If some non-particle regions are close to particle regions, which may also be marked out. Additionally, our method only marks out particle-like regions, and could be disturbed by a more fuzzy data.

5. Conclusion

Our method trained a cGAN model with synthetic images as training set and successfully mark out particle regions on real micrographs. We achieved the state of the art at fully automatic particle picking method. Our method showed robust results on different datasets.

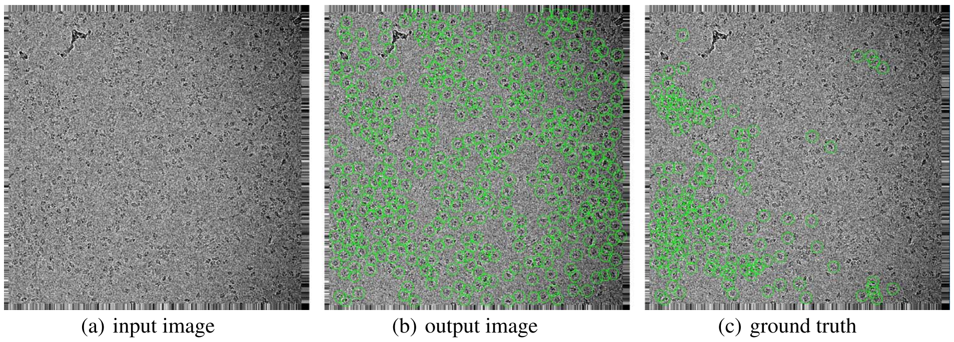


Figure 9: **Bad particle picking results on EMPIAR-10017 dataset (with a precision of 0.22 and a recall of 0.76).** Comparing (b) with (c), we find that many regions appear to contain particles. However, these regions are not shown in coordinates given by the dataset. Particles are highlighted with green circles according to the input coordinate file.

References

[1] Yifan Cheng, Nikolaus Grigorieff, Pawela. Penczek, and Thomas Walz. A primer to single-particle cryo-electron microscopy. *Cell*, 161(3):438–449, 2015.

[2] Yan Chuangye, Hang Jing, Wan Ruixue, Huang Min, Catherine C L Wong, and Shi Yigong. Structure of a yeast spliceosome at 3.6-angstrom resolution. *Science*, 349(6253):1182–1191, 2015.

[3] Liao Maofu, Cao Erhu, Julius David, and Cheng Yifan. Structure of the trpv1 ion channel determined by electron cryo-microscopy. *Nature*, 504(7478):107, 2013.

[4] William V. Nicholson and Robert M. Glaeser. Review: Automatic particle detection in electron microscopy. *Journal of Structural Biology*, 133(2-3):90–101, 2001.

[5] Joseph A. Mindell and Grigorieff Nikolaus. Accurate determination of local defocus and specimen tilt in electron microscopy. *Journal of Structural Biology*, 142(3):334–347, 2003.

[6] David W. Andrews, Alex H. C. Yu, and F. Peter Ottensmeyer. Automatic selection of molecular images from dark field electron micrographs. *Ultramicroscopy*, 19(1):1–14, 1986.

[7] Robert Langlois, Jesper Pallesen, Jordan T. Ash, Danny Nam Ho, John L. Rubinstein, and Joachim Frank. automatic particle picking for low-contrast macromolecules in cryo-electron microscopy. *Journal of Structural Biology*, 186(1):1–7, 2014.

[8] James Z. Chen and Grigorieff Nikolaus. Signature: a single-particle selection system for molecular electron microscopy. *Journal of Structural Biology*, 157(1):168–173, 2007.

[9] Joachim Frank and Terence Wagenknecht. Automatic selection of molecular image from electron micrographs. *Ultramicroscopy*, 12(3):169–175, 1983.

[10] Steven J. Ludtke, Philip R. Baldwin, and Wah Chiu. Eman: Semi-automatic software for high-resolution single-particle reconstructions. *Journal of Structural Biology*, 128(1):82, 1999.

[11] Sjors H. W. Scheres. Relion: Implementation of a bayesian approach to cryo-em structure determination. *Journal of Structural Biology*, 180(3):519–530, 2012.

[12] Ali Punjani, John L. Rubinstein, David J. Fleet, and Marcus A. Brubaker. cryosparc: algorithms for rapid unsupervised cryo-em structure determination. *Nature Methods*, 14(3):290, 2017.

[13] P. S. Adiga, R. Malladi, W. Baxter, and R. M. Glaeser. A binary segmentation approach for boxing ribosome particles in cryo em micrographs. *Journal of Structural Biology*, 145(1):142–151, 2004.

[14] Adil Al-Azzawi, Anes Ouadou, John J Tanner, and Jianlin Cheng. Autocryopicker: an unsupervised learning approach for fully automated single particle picking in cryo-em images. *BMC Bioinformatics*, 20(1):326, 2019.

[15] Ayelet Heimowitz, Joakim Andn, and Amit Singer. Apple picker: Automatic particle picking, a low-effort cryo-em framework. *Journal of Structural Biology*, 204(2):215–227, 2018.

[16] Marti A. Hearst. Support vector machines. *IEEE Intelligent Systems & Their Applications*, 13(4):18–28, 1998.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *International Conference on Neural Information Processing Systems*, 2012.

[18] Feng Wang, Huichao Gong, Gaochao Liu, Meijing Li, Chuangye Yan, Tian Xia, Xueming Li, and Jianyang Zeng. Deeppicker: A deep learning

approach for fully automatic particle picking in cryo-em. *Journal of Structural Biology*, 195(3):325–336, 2016.

[19] Yanan Zhu, Qi Ouyang, and Youdong Mao. A deep convolutional neural network approach to single-particle recognition in cryo-electron microscopy. *BMC Bioinformatics*, 18(1):348, 2017.

[20] Yifan Xiao and Guangwen Yang. A fast method for particle picking in cryo-electron micrographs based on fast r-cnn. In *AIP Conference Proceedings*, volume 1836, page 020080. AIP Publishing, 2017.

[21] Nguyen P Nguyen, Ilker Ersoy, Tommi White, and Filiz Bunyak. Automated particle picking in cryo-electron micrographs using deep regression. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2453–2460. IEEE, 2018.

[22] Thorsten Wagner, Felipe Merino, Markus Stabrin, Toshio Moriya, Claudia Antoni, Amir Apelbaum, Philine Hagel, Oleg Sitsel, Tobias Raisch, Daniel Prumbaum, et al. Sphire-cryolo is a fast and accurate fully automated particle picker for cryo-em. *Communications Biology*, 2(1):218, 2019.

[23] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017.

[24] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 2019. MR3796894

[25] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *Computer Science*, pages 2672–2680, 2014.

[26] Phillip Isola, Jun Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. 2016.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. 2015.

[28] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. 2016.

[29] Marin Van Heel. Detection of objects in quantum-noise-limited images. *Ultramicroscopy*, 7(4):331–341, 1982.

[30] Andrii Iudin, Paul K. Korir, José Salavert-Torres, Gerard J. Kleywegt, and Ardan Patwardhan. Empiar: a public archive for raw electron microscopy image data. *Nature Methods*, 13(5):387, 2016.

[31] Sjors H. W. Scheres. Semi-automatic selection of cryo-em particles in relion-1.3. *Journal of Structural Biology*, 189(2):114–122, 2015.

[32] Radostin Danev and Wolfgang Baumeister. Cryo-em single particle analysis with the volta phase plate. *eLife*, 5, 2016.

Fang Kong
School of Information
Renmin University of China
Beijing 100872
China
E-mail address: 2017104078@ruc.edu.cn

Xirong Li
School of Information
Renmin University of China
Beijing 100872
China
E-mail address: xirong@ruc.edu.cn

Qing Liu
School of Information
Renmin University of China
Beijing 100872
China
E-mail address: qliu@ruc.edu.cn

Chuangye Yan
School of Life Sciences
Tsinghua University
Beijing 100091
China
E-mail address: yancy2015@mail.tsinghua.edu.cn

Xinqi Gong
Institute for Mathematical Sciences
Renmin University of China
Beijing 100872
China
E-mail address: xinqigong@ruc.edu.cn