Final Project

Name: Aris Podotas

University: National and Kapodistrian University of Athens

Program: Data Science and Information Technologies

Specialization: Bioinformatics - Biomedical Data

Lesson: Algorithms In Structural Bioinformatics

Date: June 23, 2025

Contents

1 Abstract

Cryo Electron Microscopy particle picking is still a method that does not output the necessary accuracy, or more importantly recall needed. Various machine learning frameworks have been applied but there is no one solution that would produce the best metrics to set an industry standard yet. In order to keep up with the pharmaceutical need for new and effective drugs more protein structures need to be identified and specifically of proteins that do not fit well into the methodology of crystallization as other methods aside from the Cryo-EM rely on. Here we attempt to better the Cryo-EM particle picking problem using un-supervised classification by proposing novel methods.

2 Introduction

2.1 Cryo-EM

Cryo Electron Microscopy (often abbreviated Cryo-EM) is a crucial method for identifying protein structures Banerjee et al., 2020, that holds major benefits over other methods in specific circumstances. One major benefit of the Cryo-EM method is that the protein in question does not need to be crystallized before the procedure Chari and Stark, 2023. Crystallization is a prerequisite for other alternative methods. If there is a crystallized prerequisite step in a method, then it will exclude potential proteins from being structured since they do not crystallize Chari and Stark, 2023. Cryo-EM as a method relies on a laboratory experiment part and a informatics based analysis of the results for the generation of the final structure prediction, commonly these two steps are referred to as the wet and dry lab respectively.

The main process behind the wet lab component is to isolate the protein in some purified solution using chromatography for instance. This process should keep the protein intact and the concentration of the protein in the final solution should be within some standard. A small circular grid, specific for these experiments is prepared for adding the solution containing the protein, once coated with solution the grid or micrograph is frozen. Ideally the instances of the protein would remain intact through this process and the solution in question would be such that it does not cause unfolding or denaturing of the sample.

The main process for the dry lab component is to scan the micrograph for imaging of the area of the whole platelet for processing and identification of the molecule positions inside the images, and thus over the area of the micrograph itself in total. Grey-scale images of consistent size are taken over the area of the micrograph. Each image will contain two dimensional slices of instances of the protein in question in different orientations called particles. Particle picking is the utilization of some pipeline ranging from a unsupervised classification Li, 2022 to a traditional supervised classifier Al-Azzawi et al., 2019a, Al-Azzawi et al., 2020, Al-Azzawi et al., 2019b and even deep learning methods like convolutional neural networks Al-Azzawi et al., 2020, Chung et al., 2022, Xu et al., 2024 can be used to find the locations of the particles in the images. All of the pipelines include some image processing to aid the particle picking process as much as possible in de-noising the images. The original images of the scanned micrograph are of grainy quality, requiring some pre-processing to happen over all the images. Traditionally image processing for de-noising is done with the use of some

edge detection method such as the Laplacian or the fast Fourier transform Ovall, 2016 Brigham and Morrow, 1967.

Here, we focus on the pipeline of the dry lab and the particle picking problem using unsupervised classification or clustering with image processing methods.

3 Methods

3.1 Implementation

All results were generated in the Python programming language "3.13.2 Documentation", n.d. Libraries include

1. OpenCV "OpenCV: OpenCV modules", n.d.
2. Numpy "NumPy", n.d.
3. Optuna "Optuna - A hyperparameter optimization framework", n.d.
4. Matplotlib "matplotlib.axes — Matplotlib 3.10.1 documentation", n.d.
5. Sklearn "scikit-learn: machine learning in Python — scikit-learn 1.6.1 documentation", n.d.
6. The chameleon clustering algorithm Lin, 2025

All implementation files can be found in the zip file containing this report in the ./src/ directory.

All scripts are meant to be ran in the ./src/ directory with the default paths, should one want to run the pipeline from a project that has a different architecture and outside the ./src/ directory they should change the optional paths in the command line input. The naming scheme for all the files follows the following scheme:

\d*_\d*.png

Saving functionality and loading functionality has been implemented.

3.2 Data

Images were provided by the Athena research center, Andreas Zamanos Zamanos et al., n.d. A total of $1639 + 545 + 550$ images (train, validation, test sets in that order) were provided, each image with resolution 1024×1024.

3.3 Image Processing

For each image, a series of images was generated. The list of transformations is as follows:

1. The Laplacian of the image Ovall, 2016
2. Filtering all pixels with value less than x
3. Filtering all pixels with value more than x
4. Filtering a range of values of pixels (recursive less than x and more than y calls)
5. Conversion of each pixel with values x to y

Then, on the any of these images a series of filters can be generated for nested combinations of filters. Filtering refers to setting hit values to $(0, 0, 0)$. Images in-between steps can optionally be saved or otherwise only kept as class attributes. Not all of these transformations are useful for each image, specific filters are best

for each image. Each filter is saved with the last transformation coming first in the file name (after the name of the original image). The purpose of these transformations is to make the data suitable for clustering in a different format, using pixel coordinates as feature vectors for all pixels that have not been set to black $(0, 0, 0)$; instead of using color channel features.

The default filter in the pipeline uses the filtering of all pixels outside a narrow range $(69, 74)$, then taking the Laplacian matrix Ovall, 2016. The default kernel is:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

If one is to change the kernel they should make sure the entries in the matrix sum to 0.

A more analytical definition of the filters (except the Laplacian Ovall, 2016) would be:

The "Less than" filter,

$$f(x, y) = \begin{cases} x & \text{If } x \leq y \\ 0 & \text{Otherwise} \end{cases}$$

The "More than" filter,

$$f(x, y) = \begin{cases} x & \text{If } x \geq y \\ 0 & \text{Otherwise} \end{cases}$$

The "Range" filter,

$$f(x, y, z) = \begin{cases} x & \text{If } z \leq x \leq y \\ 0 & \text{Otherwise} \end{cases}$$

The "Convert" filter,

$$f(x, y, z) = \begin{cases} z & \text{If } x = y \\ x & \text{Otherwise} \end{cases}$$

Optional histograms of each image along with the value of the average pixel along all columns, and rows will also be generated. In future additions the defaults will be defined from the histogram itself.

3.4 Labels

Labels were presented as .star files containing images and coordinates of each identified molecule structure within the image.

If labels are to be provided (since the script will work on any image without labels as an independent clustering pipeline) they should be in the folder specified by the command line and they should be .star files that follow the naming scheme of the images, using the first integer before the _ character ((\d*)_\d*.png, the first capture group 3.1) as the identifier of the star file with the labels (so
d*.star). They should contain independent lines of the image name and the coordinates of each molecule with the columns being separated by spaces. Integers and floating point numbers will work in the pipeline.

A parser was written (./src/starParser.py) so that the labels of the images can be parsed from the .star files into Python "3.13.2 Documentation", n.d. All files available in the zip containing this assignment.

3.5 Clustering

The following clustering algorithms were used:

1. K means
2. DBSCAN
3. Gaussian Mixture Model Expectation Maximization

The intent was to use all of:

1. K means
2. DBSCAN
3. Gaussian Mixture Model Expectation Maximization
4. The chameleon
5. Valley seeking algorithms
6. Competitive Learning

However, the Chameleon relies on the C programming language Kernighan and Ritchie, 1988 and the Valley seeking, Competitive Learning have no found implementation at all.

Other algorithms were imported from sklearn "scikit-learn: machine learning in Python — scikit-learn 1.6.1 documentation", n.d.

The assumptions that lead to these choices of algorithms were

1. Clustering algorithms that capture abstract shapes will perform better
2. Algorithms that are mode seeking will perform well since the number of clusters is variable Cheng, 1995
3. Fast algorithms will be preferred because of the volume of the data

3.6 Metrics

The following metrics were used:

1. Accuracy score,
2. Recall score,
3. F1 score,
4. Fbeta score,
5. Precision score,
6. Jaccard score "Jaccard Similarity - an overview — ScienceDirect Topics", n.d.

In no particular order.

Metrics were imported from sklearn "scikit-learn: machine learning in Python — scikit-learn 1.6.1 documentation", n.d.

The fbeta score was used with a beta value of $b = 2$ for an emphasis on recall. The emphasis on the recall score was great in the assignment as this is the most key point and the hardest challenge in the particle picking process overall. The Jaccard score differs in nature from the other metrics for it will signify a quality of predictions and not a predictions state (True, False).

3.7 Label Mapping

Labels were given as coordinates as said previously 3.4. At the same time we are using classification metrics 3.6, meaning that our labels and predictions should be some form of binary-multi class labels.

These two facts about the project mean that some conversion or mapping needs to be applied to the coordinates unless we are to use the set of 1000 labels the coordinates would represent (which would be an error).

There are two paradigms present in the assignment and a comparison between them will be facilitated.

The first paradigm (custom grid): The conversion was done using a grid over the image where if a cell in the $2D$ grid contains a labels coordinates or a predictions

coordinates then a new array that represents the labels or the predictions accordingly will be padded with a 1 value, otherwise 0 values are padded. This is similar to the way the perithlasigram is parsed in X-ray crystallography. The grid size is a variable but the chosen value after testing is 100 pixels per grid cell which maps to the size of the instances of the molecule.

Warning: If a small enough grid cell size is chosen the label array generated will have a great amount of 0 labels and as a result all metrics with false predictions on the denominator will tend to the form $\frac{x}{0}$.

The second paradigm (imported distance): The conversion is based on a check over all the pairs of labels and predictions, keeping 1 labels if the distance should be under a specific threshold (imported from Andreas Zamanos).

The first grid based approach has the benefit of being fast and memory efficient but there are ideological consideration like the size of the grid changing the metrics that hold it back from being the optimal solution overall.

3.8 Plotting

All plots were made in python using OpenCV "OpenCV: OpenCV modules", n.d. and matplotlib "matplotlib.axes — Matplotlib 3.10.1 documentation", n.d.

3.9 Feature Extraction

Once a filtered image in the proper format has been made (see below for what constitutes a proper format) the extraction of the points (pixels) as if the image was a scatter plot is facilitated. This is the novel approach for this assignment as alternative methods focus on Intensity Based Clustering (IBC for short).

A proper format for an image is one where all pixels are gray scale and pixels to not be considered as valid points are set to the value $0, 0, 0$ or black.

The extraction of the points is done by keeping an array of the indexes (pixel row and column) of each non $0, 0, 0$ pixel.

In a sense this method is to convert the image to a scatter plot where points exist in a two dimensional plane (rows and columns of an image). This method heavily relies on the image processing being able to identify the molecules via edge detections.

Clustering algorithms were adapted to this format of the data so that the results can be drawn back to the images. This format has allowed otherwise useless algorithms to be applied to the particle picking problem. This method is slower than the traditional IBC. Main benefits to this approach over traditional methods is the intrinsic noise reductions to the method stemming from the removal of pixels (setting to 0 values). This is something that the IBC cannot do since the clustering space is of the pixel values (not positions) and as such all the pixels in one cluster with the same value must be kept only if a whole cluster is removed will those points be removed. For example the DBSCAN, Chameleon algorithms are otherwise not useful to run and hard to interpret in IBC, leading to outputs that cluster most of the image together since the clusters along the $[0, 255]$ axis of pixel values do not separate well.

Most image processing methods 3.3 are then not novel but their application in this

framework is of a novel use case

3.10 Feature Selection

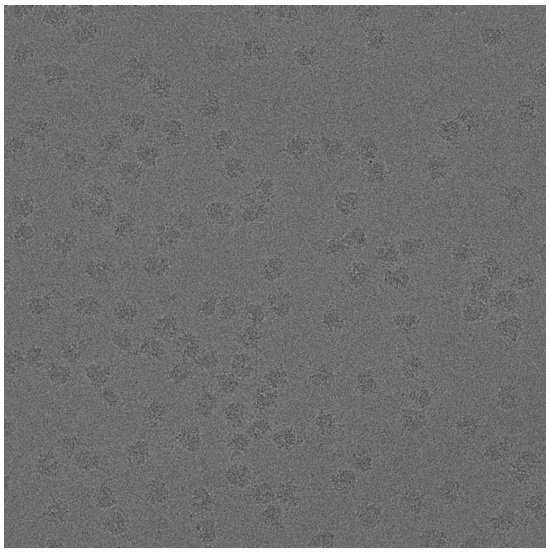
Since the clustering is based on the locations of the pixels and not the color channels the image essentially represents a scatter plot and the goal is to get only the pixels of the instances of the molecule to be non black $(0, 0, 0)$. In this framework there is some feature selection to be done, in that the remaining pixels are:

1. Grayscale
2. 3-tuples

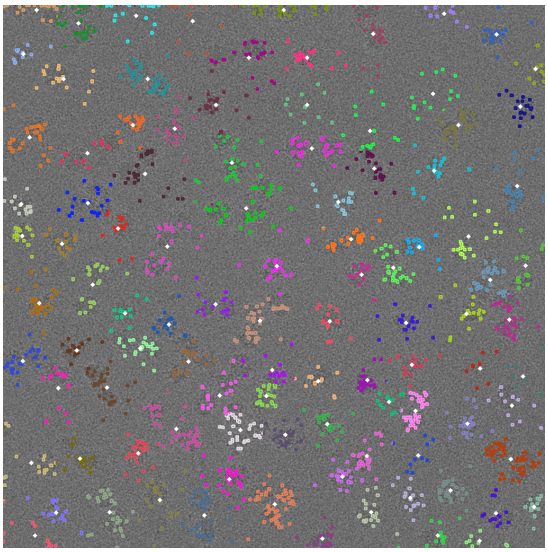
The feature selection to do then is to remove the two of the three color channels and keep only the remaining one with the value all 3 had. This had major improvements of the runtime of the algorithms.

4 Results

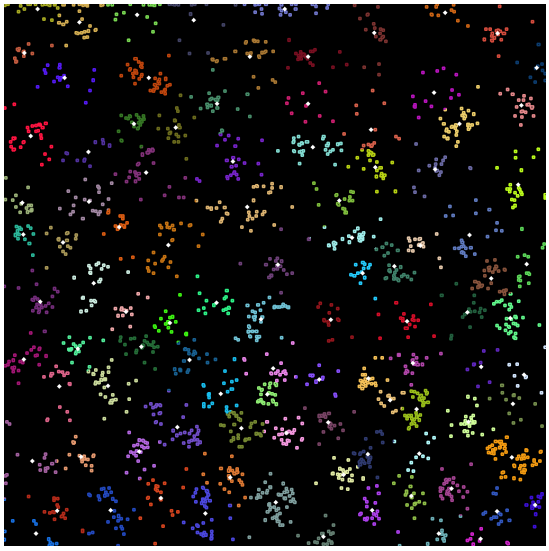
4.1 K-Means



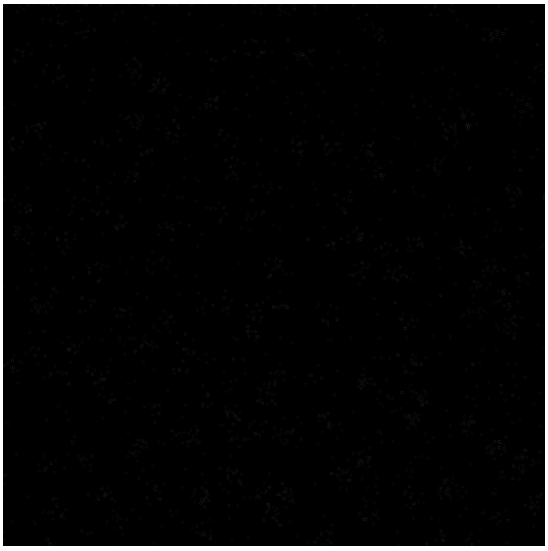
(a) Original



(b) Clustering Overlap



(c) Predictions only



(d) Edge detection

Figure 1: Example of k-means outputs. Top left: The original image scanned from the micrograph directly. Top right: The outputs of the clustering algorithms overlapped on the original, white points are cluster representatives and colored points are data vectors within the cluster (segregated by color). Bottom left: Only the representatives and their predictions of the vectors following the same scheme as the top right. Bottom right: Image after preprocessing fed to the clustering algorithm. The edge detection is hard to see since the background of the page is white but upon zooming in the points are visible, it is not empty. The labels are of the mean values

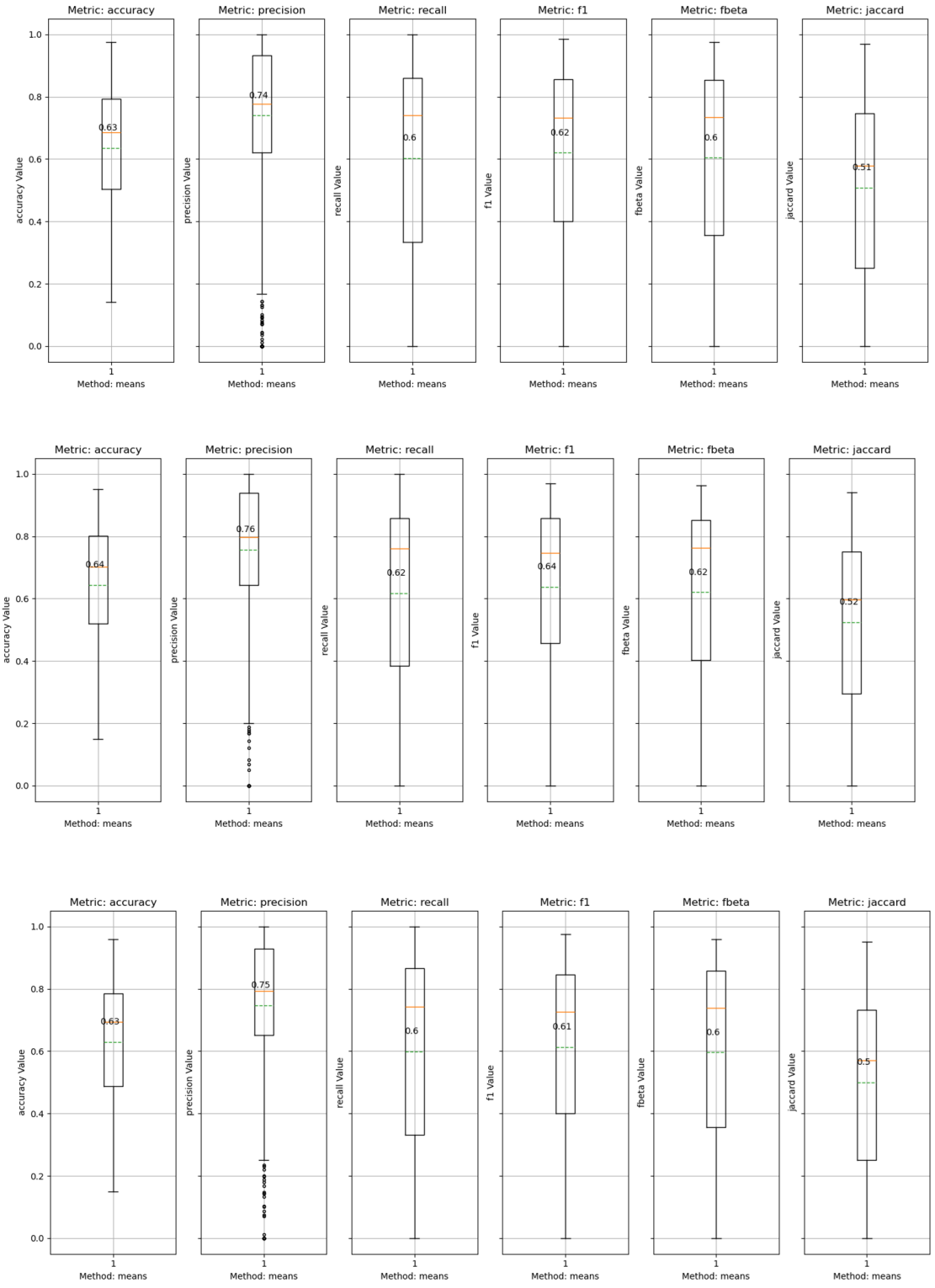


Figure 2: K-means overall evaluation, in order Train-Validation-Test. The green dotted line denotes the mean value and the orange one the median value. The labels are of the mean values

4.1.1 Results

Looking at image 1 we see that the clusters themselves have sometimes captured a single particle and sometimes split between particles and less often captured multiple particles. We can see that points that should be considered noise are captured by the algorithm.

Looking at image 2 we see that the distributions have a high variance, this seems to be a direct results of outliers considering the way the median value is on average 10 points higher than the mean value. Looking at median values as a result of understanding this we see that most metrics lie in the $[70, 80]$ range. It seems that no significant difference exists between the different sets of data (train, validation, test). This is not surprising considering that unsupervised classification is not a method that utilizes training, validating and testing. There are also too many images input with too great a variance evident by the spanning of the whole metrics space of the distributions.

4.1.2 Negative Remarks

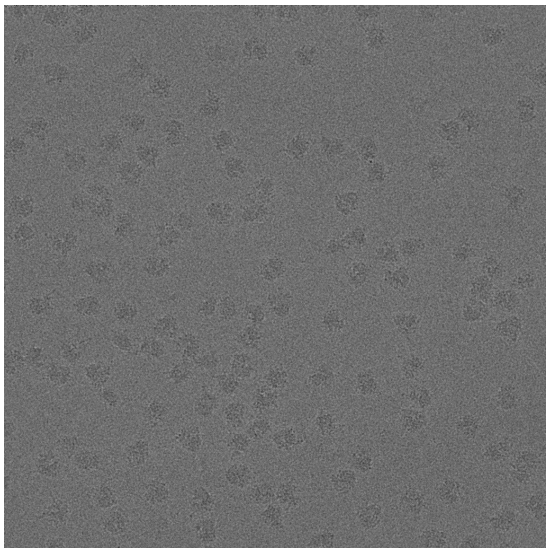
Specific comments about the evaluations as a whole are offered in the 4.4 section. The k-means is an algorithms that requires it's cluster number or n to be given beforehand, a protocol to identify the number of cluster exists however in this assignment it was not completed to any standard that could have been used for each image. A preset number was used where images will really low numbers of pixels would default to having as many clusters as the number of pixels to avoid any errors. The difference between the real number of clusters and the preset should reduce metrics. Despite this the metrics and especially the recall score are better than random and one has to wonder what a dynamic n search would yield instead. Specific emphasis was given to the recall score in the assignment and the results show that the recall score is the point that the algorithms struggle with the most. The recall has the biggest variance of the predictions and has affected the F1 score and Fbeta score. The positive predictions are simply more accurate than the negative ones.

The biggest concern from the metrics is the Jaccard score "Jaccard Similarity - an overview — ScienceDirect Topics", n.d., which is 50% and would indicate that when a cluster has captured a molecule it only gets about half of the molecule and not the majority of the shape. This would be the biggest issue since the steps after the particle picking are to generate sub images of the instances of the protein in the micrograph and a small overlap would yield a low quality set of images and thus a bad overlap and coverage of the three dimensional molecule - structure. These metrics aside form the Jaccard score do not differ significantly from industry standard Zamanos et al., n.d., Dhakal et al., 2024 so far as the median value is concerned. Some industry results have been omitted from comparison for they are not in the same format (data points, not distributions of outputs).

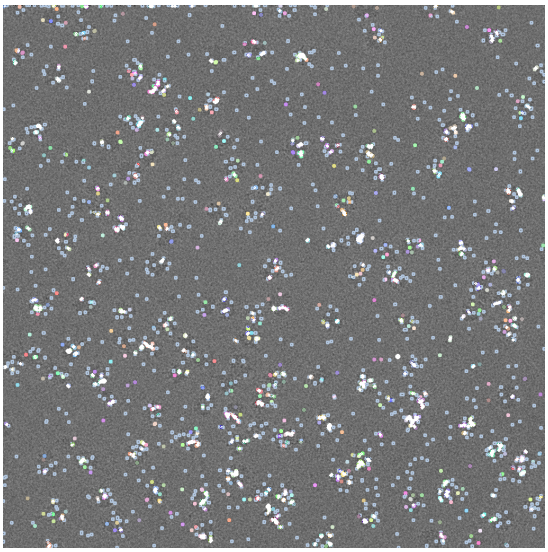
One last point to consider is that the k-means doesn't label any points as noise and as such the positions of the predictions will be affected by any and all noise in the image left after the image processing.

It is expected that the algorithms other than the k-means should perform better. This will be elaborated on in the following chapters.

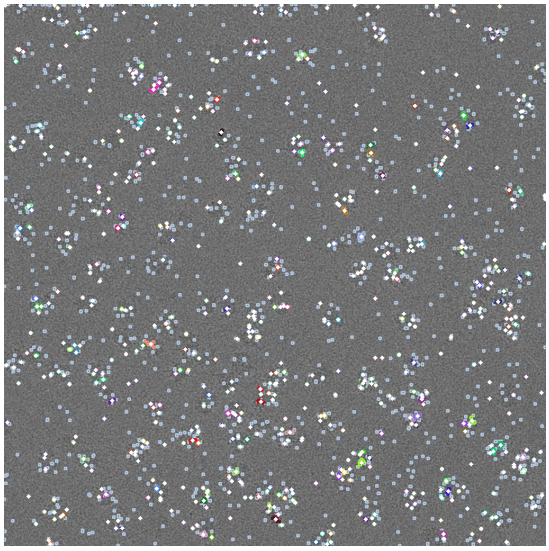
4.2 DBSCAN



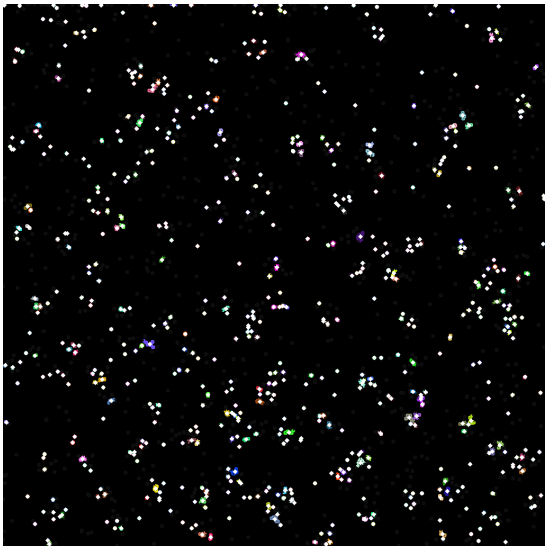
(a) Original



(b) Clustering Core Sample Overlap



(c) Cluster Representatives Overlap



(d) Predictions Only

Figure 3: Example of DBSCAN outputs. Since this is the same original image we do not give the image processing edge detection again. Top left: The original image scanned from the micrograph directly (same as the k-means 1). Top right: The outputs of the clustering algorithms overlapped on the original, white points are core samples and colored points are data vectors within the cluster (segregated by color). Bottom left: The outputs of the clustering algorithms overlapped on the original, white points are cluster representatives and colored points are data vectors within the cluster (segregated by color). Bottom right: Only the predictions of the algorithms following the representative scheme.

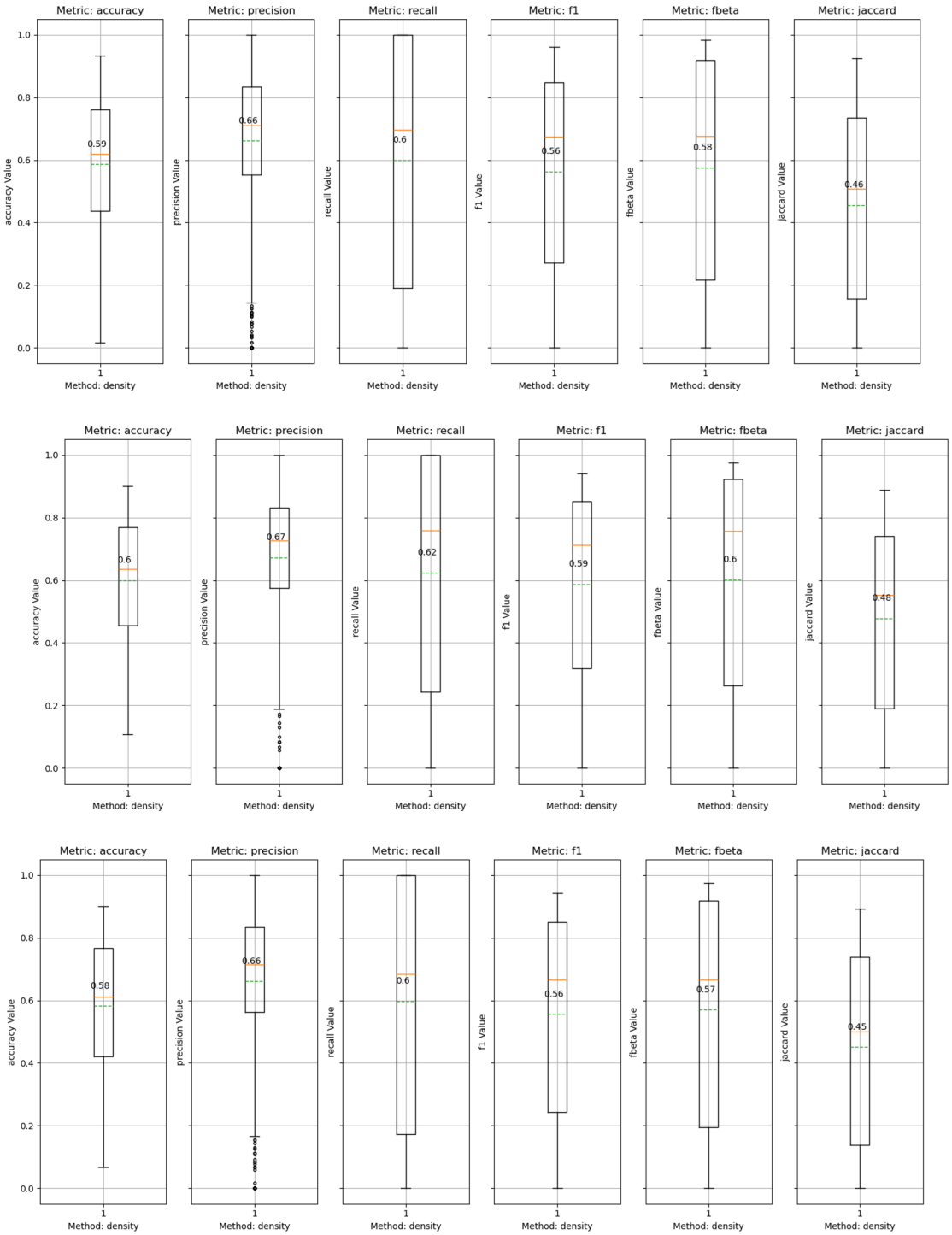


Figure 4: DBSCAN overall evaluation, in order Train-Validation-Test

4.2.1 Results

Looking at image 3 we see that the clusters themselves have captured on average less

than a full cluster, noise has accurately reduced the positive involvement of noisy points.

Looking at image 4 we see that the distributions have a high variance (the highest of our algorithms), this seems to be a direct results of the algorithm having a unique way of separating clusters and the hyper-parameters provided since the line between clustering the entire image in one cluster and clustering each point separate is very narrow. Looking at median values as a result of understanding this we see that most metrics lie in the $[70, 80]$ range (with slightly worse on average than other algorithms). It seems that no significant difference exists between the different sets of data (train, validation, test). This is not surprising considering that un-supervised classification is not a method that utilizes training, validating and testing. There are also too many images input with too great a variance evident by the spanning of the whole metrics space of the distributions.

4.2.2 Negative Remarks

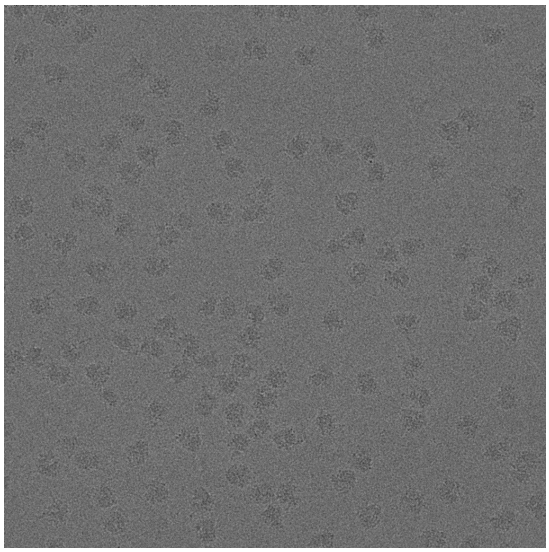
DBSCAN was considered to be an algorithms that would not yield any results for this project. The original use case for DBSCAN was so that we can capture the abstract shapes of the molecules themselves, since in the micrograph the particles take any orientation and any two dimensional slice of the three dimensional structure.

DBSCAN will note some points as outliers in the set and they will not be used to update the position of the molecule prediction. DBSCAN will output many small clusters around dense areas and the only dense areas in the images after the image processing should be the particles.

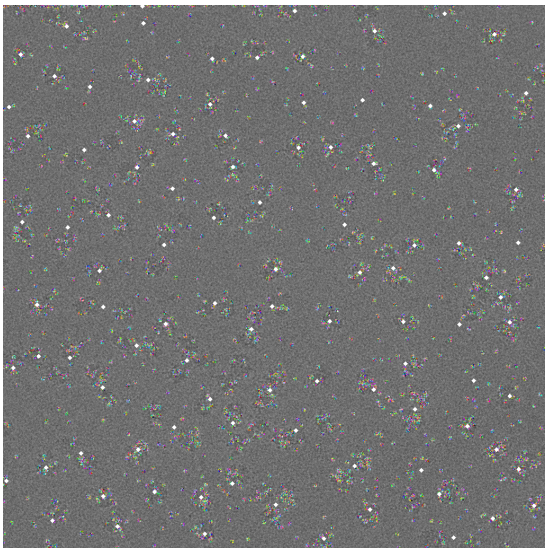
A protocol was written to generate the mean centers of the core samples from the outputs of the DBSCAN algorithm and those new points were labeled as representatives for the application of the metrics. In this framework it is predicted that DBSCAN out-performs the k-means since the exclusion of noise should simply increased predictions.

This is not what has been found however for the metrics aside from having worse or equal variance, also have lower mean values. The only positive outcome of DBSCAN is that the recall score was able to reach a value of 1 for enough of the images that it lies within the 3 standard deviation range of the distribution.

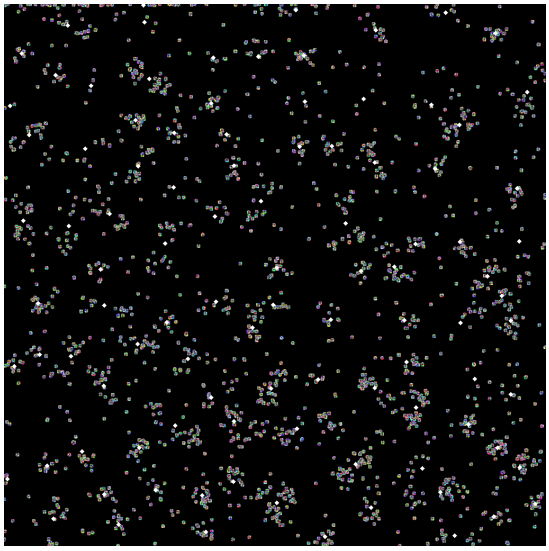
4.3 Expectation Maximization



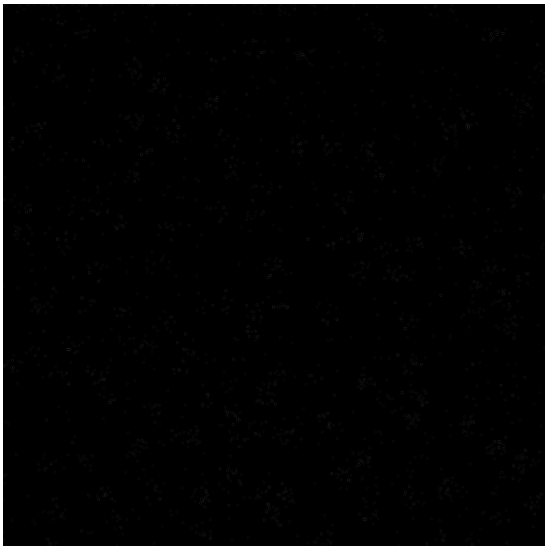
(a) Original



(b) Clustering Overlap



(c) Predictions only



(d) Edge detection

Figure 5: Example of Expectation Maximization outputs. Same as in figure 1

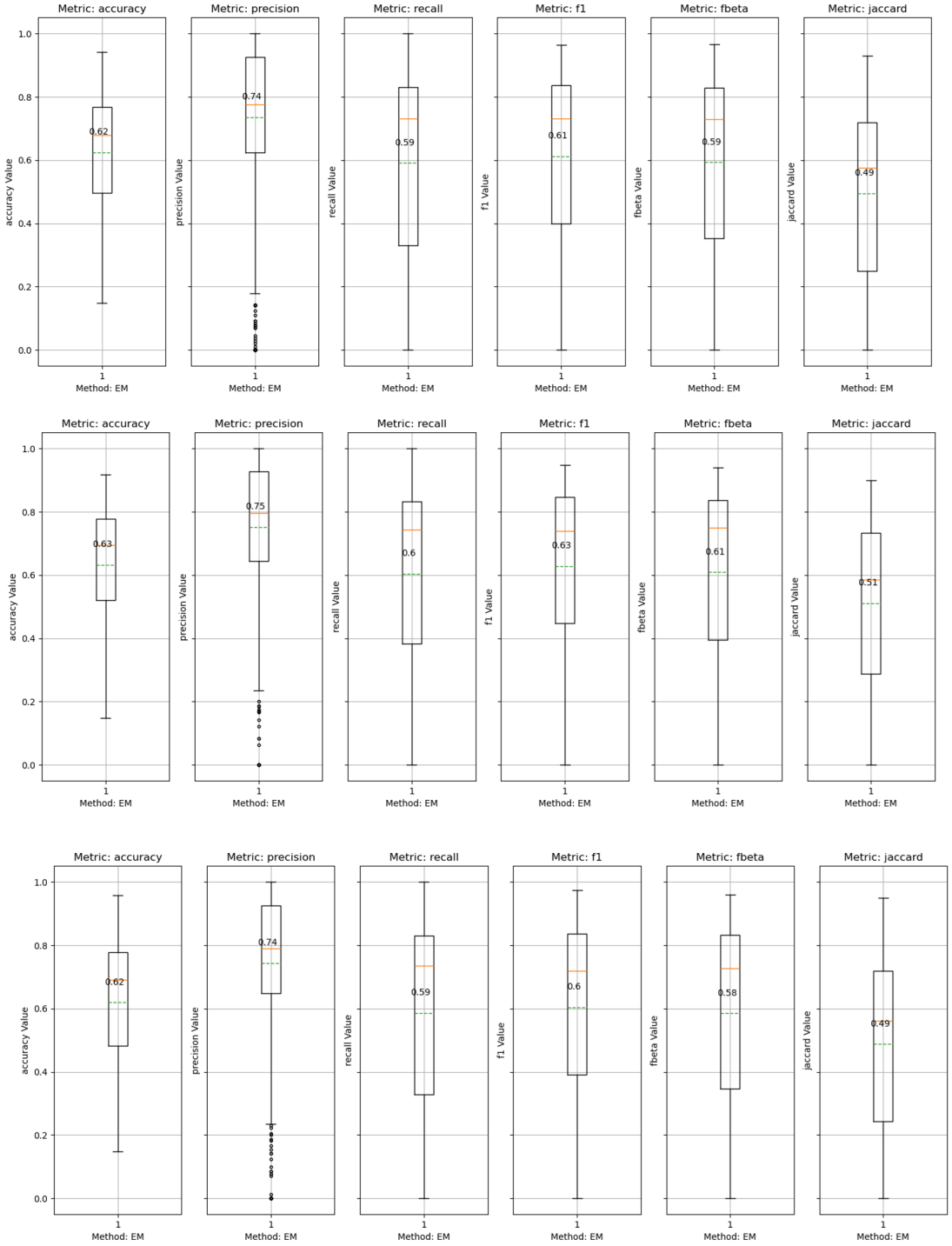


Figure 6: Expectation Maximization overall evaluation, in order Train-Validation-Test

4.3.1 Results

Looking at image 5 we see that points within any one cluster do not fit into the method that draws onto the images since each cluster seems to have a mixture of points. Otherwise the points of the rep-

resentatives have not changed much since the noise has the same likelihood as the data vectors themselves. This is a result of the image processing pipeline ran on the images and it's in ability to output noisy points with a different lightness.

Looking at image 6 we see that the distributions have are almost identical to the k-means so everything said at that sections applies here too 4.1.1.

4.3.2 Negative Remarks

The expectation maximization is simply a continuation of the k-means, as such the metrics should have improved overall. Metrics are so similar to the k-means in both variance and value that no significant difference can be noticed. The lack of difference is a result of the noisy points not being significantly different in lightness level (the likelihood used) than the points belonging to a particle .The runtime of this algorithm is slower than the k-means, meaning that if a choice was presented between them based on the metrics not differing and the runtime being skewed in favor of the k-means one should choose the k-means.

4.4 Overall

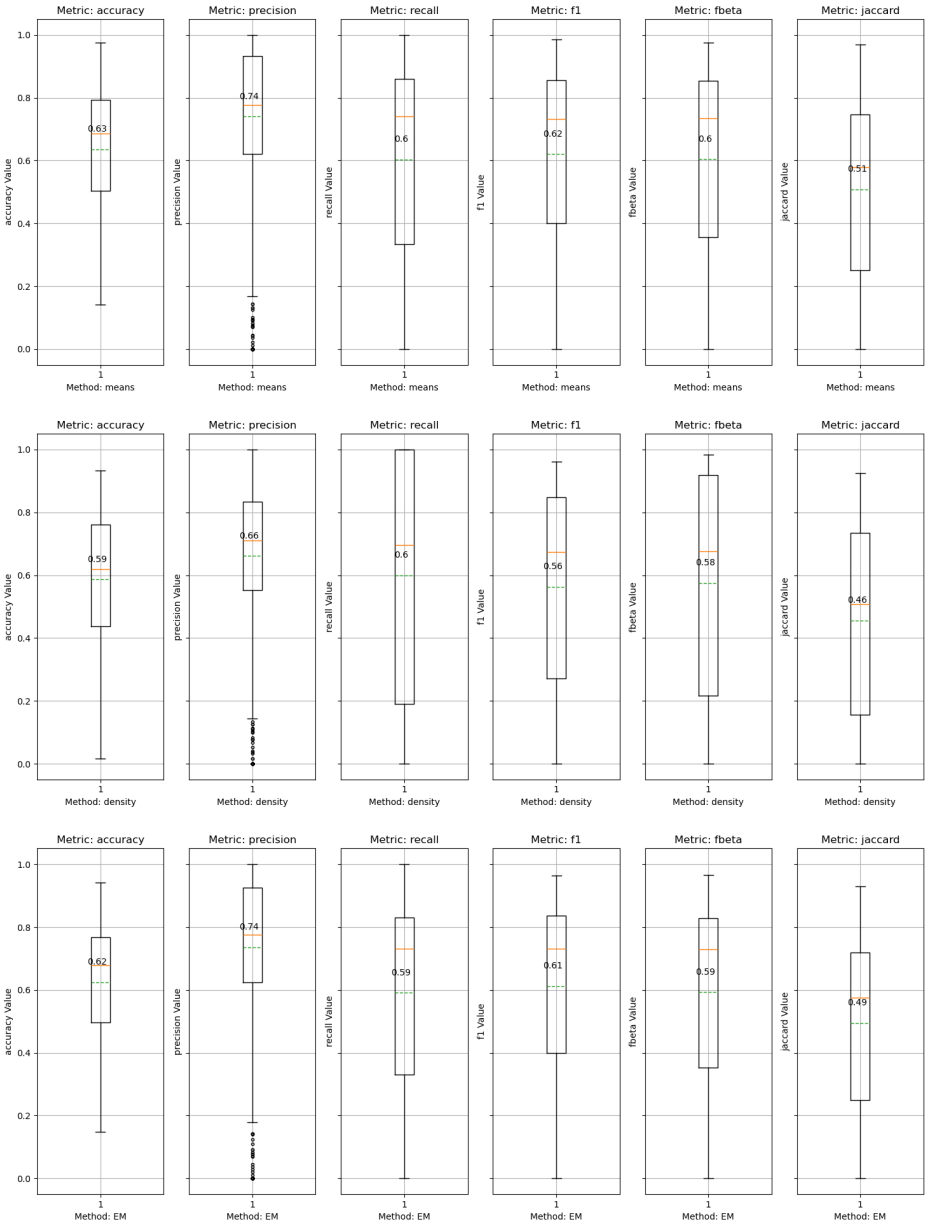


Figure 7: Overall metrics for the Training set

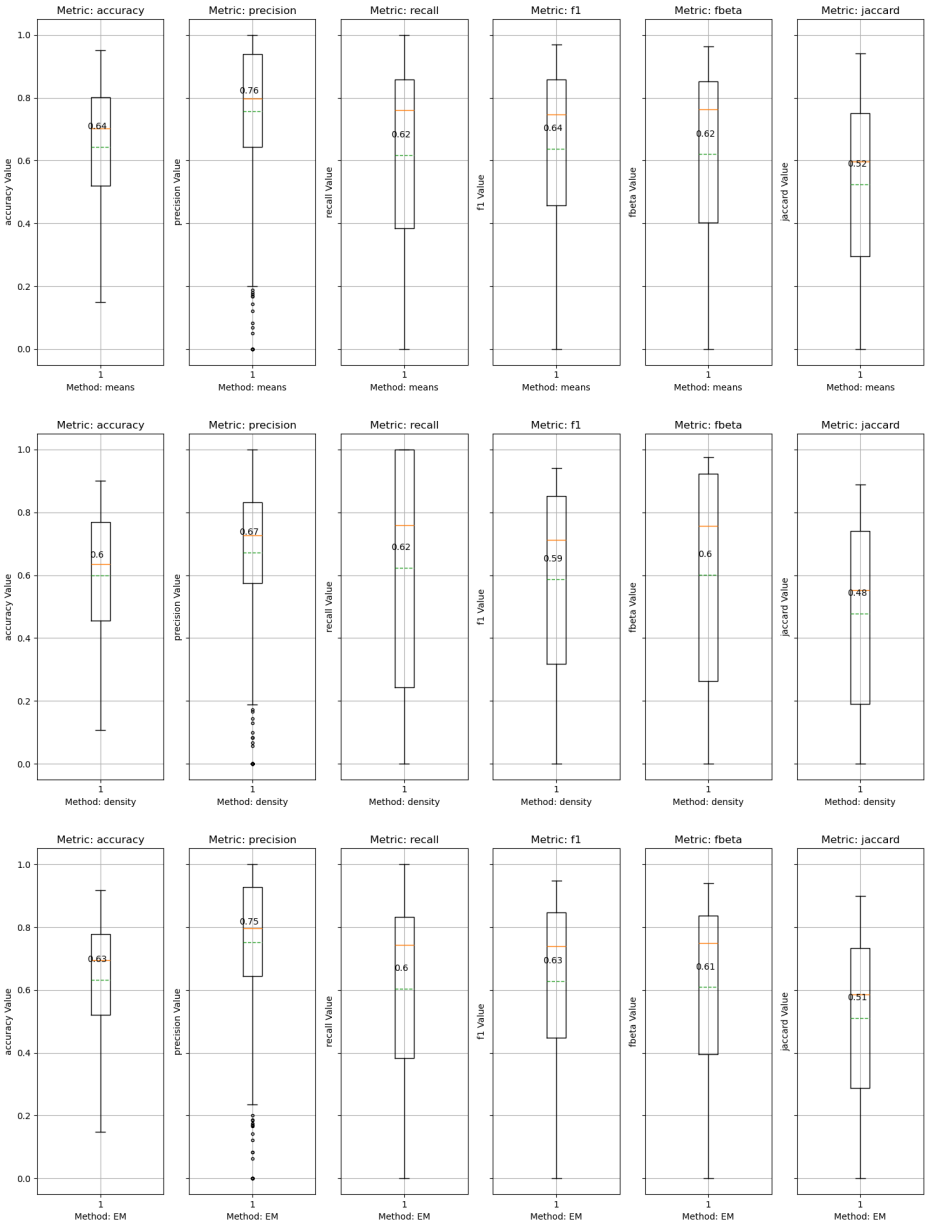


Figure 8: Overall metrics for the Validation set

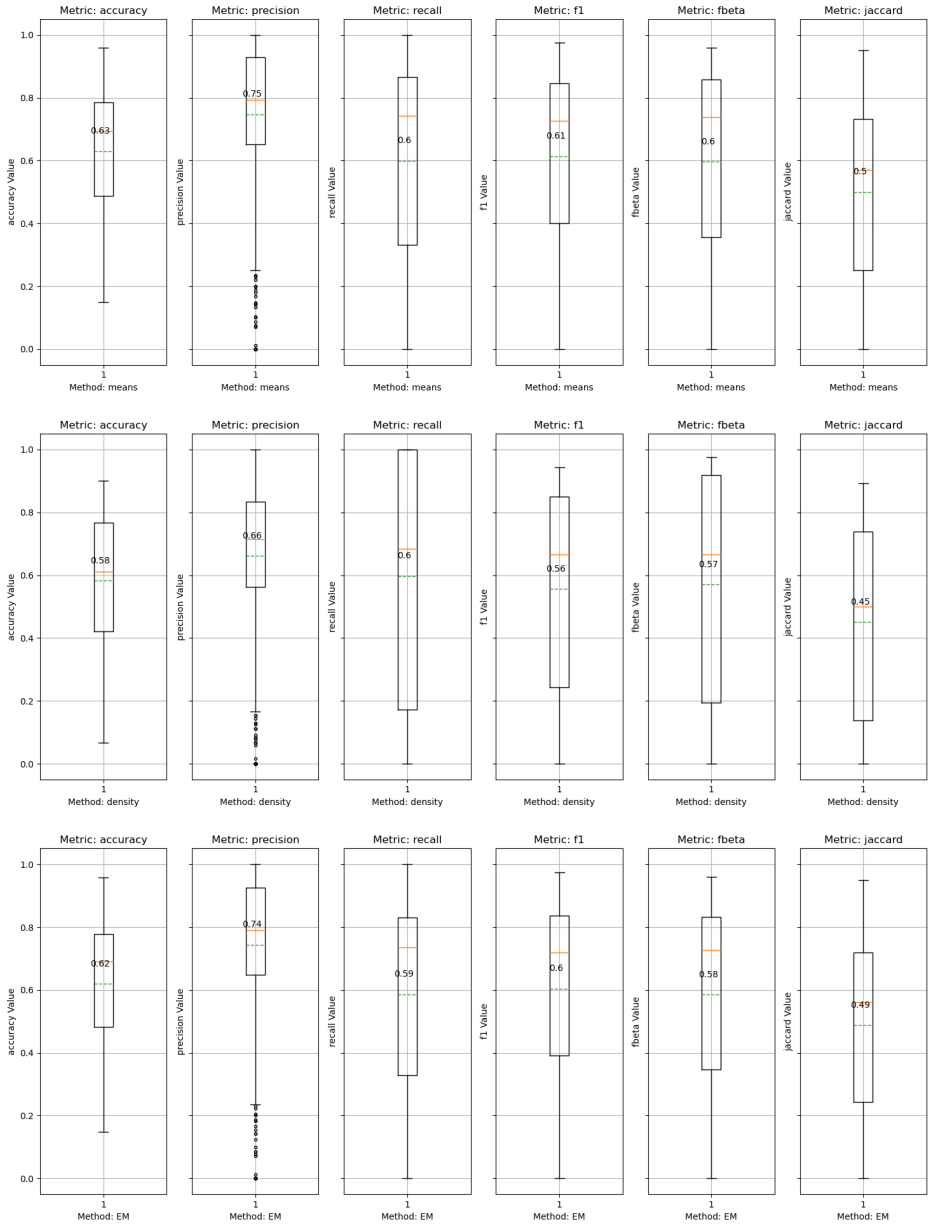


Figure 9: Overall metrics for the Test set

The indication that the mean value is always less than the median means that our data has some outliers that are affecting the results. This is also evident by the un-equal length of the confidence intervals, where the lower confidence interval is much more elongated all the way through the domain of the metric. These outliers refer to

images in the data set, examples include images like in figure 10 or 11 where there are essentially no molecules to be found.

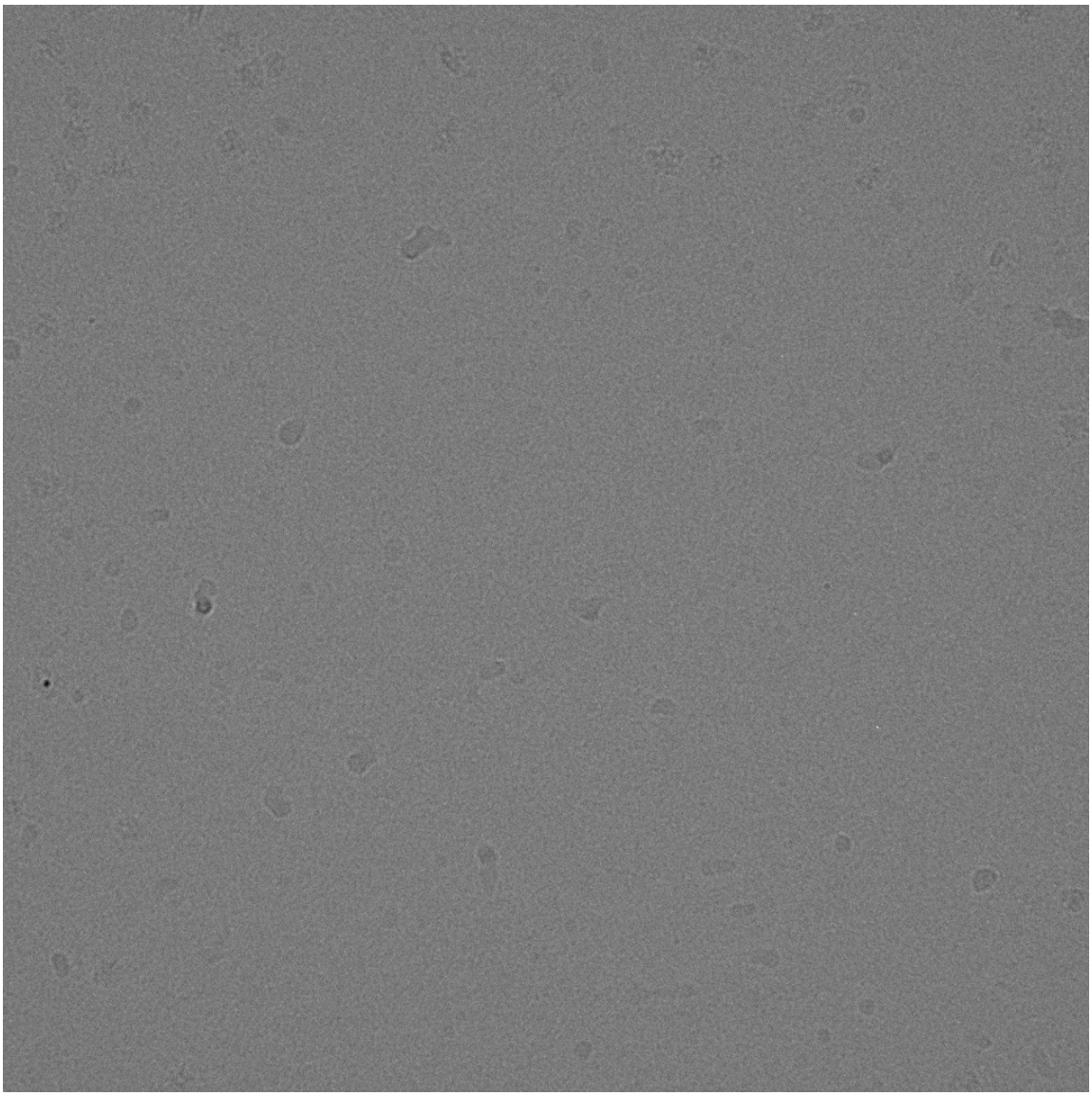


Figure 10: Example of data that has reduced the metrics and should be considered noise

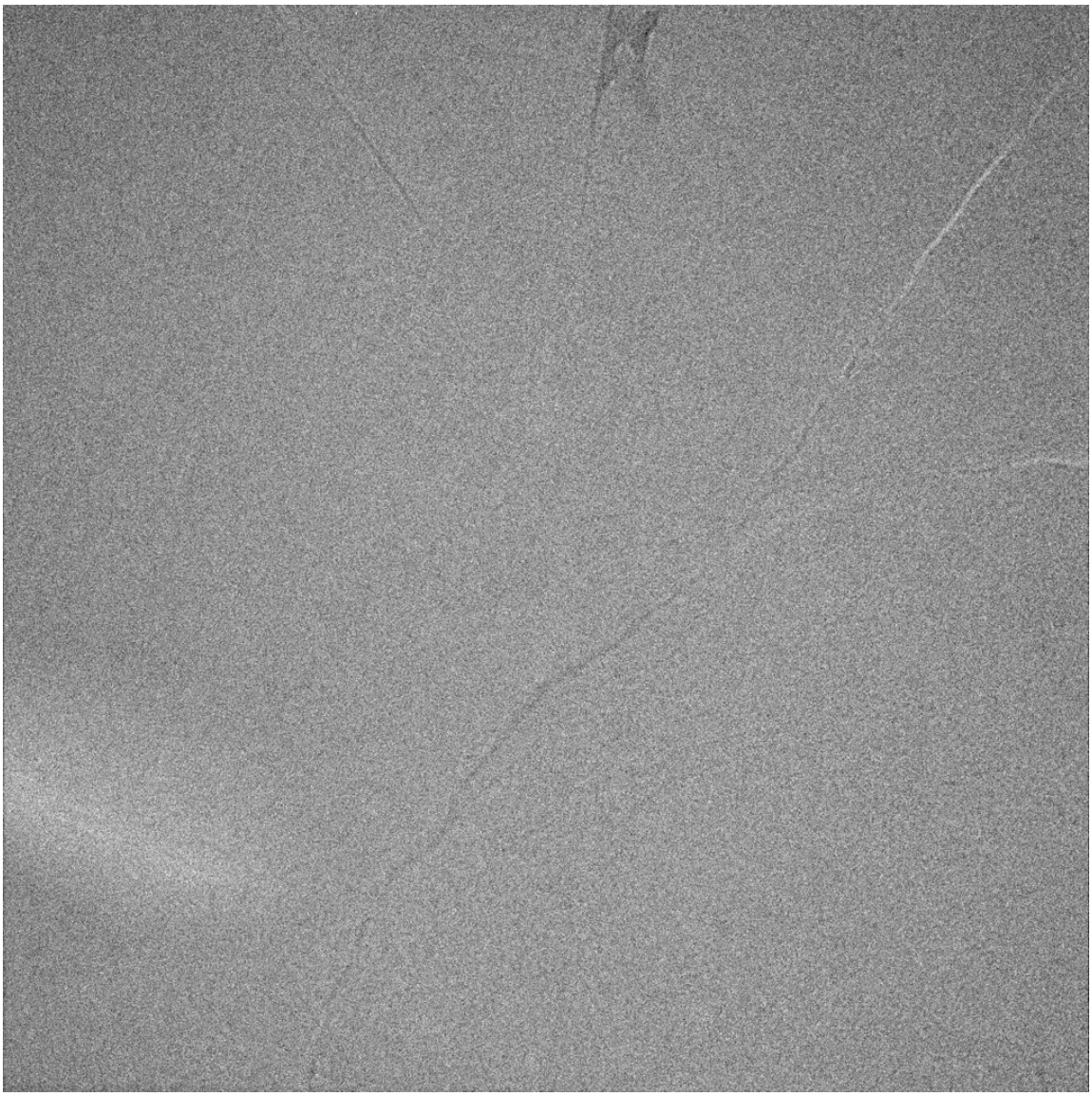


Figure 11: Example of data that has reduced the metrics and should be considered noise 2

Aside from low quality images there is the issue that the dataset contain different scales and densities of solution in different sets of images, the un-supervised classification in the non dynamic way that it is done here cannot differentiate between the resolutions and concentrations. Ideally one dataset would have only one resolution and concentration and others would be split into separate datasets that can be ran with different parameters. Once a small analysis on a specific dataset has been done the fine tuning of the current pipeline would lead to better results than it has here where all of these sets are considered using one general set of hyper-parameters. This was not found early enough in the assignment to do here.

If a un-supervised classification is to be used to make predictions about particle picking, then these issues should be resolved. Low quality images have a great affect in this scheme and it would be better if they were removed for the metrics. Unsupervised classification does not require

large data, in fact the method will perform the same on an image irregardless of training size since there is no training of the algorithms. Given this we would be better off segmenting the data into sets of similar concentration of molecule and lightness of the images and removing the split between train, test and validation sets.

One thing to note is that no significant difference can be seen between the training, validation and test set metrics. This is because the un-supervised algorithms do not have over-fitting and do not train on a dataset.

5 Conclusions

5.1 Comparison to Industry Standard

Metrics for standard approaches range from 70% to 85% for all metrics overall. Here our approach only emulates this for the precision and the positive results and not so much the negative results. More fine tuning and further work 5.2 would need to happen to approach or surpass current industry standard. The only advantage of this approach is the speed and computational ease with which results are generated that are the main driver for choosing an un-supervised classification predictor.

5.2 Further work

Further plans include:

1. Integration to the rest of the pipeline of Cryo-EM
2. Isolation of the sub images from the predictions
3. Integration with other clustering algorithms (such as the valley seeking the chameleon the competitive learning and all other clustering algorithms that are not possible to run on the novel approach we have introduced here that otherwise are uninterpretable)
4. Dynamic number of clusters integration
5. Automatic hyper parameter tuning
6. Histogram equalization
7. Integrations with other metric applying techniques
8. To find optimal grid parameter per image
9. Automatic resolutions correction
10. Dynamic pixel filtering
11. Use of the predictions and the pixels to compute a gram matrix (potentially giving an alternative way to reproduce the structure fro the Cryo-EM experiments)

To elaborate on a few of these points, the non dynamic number of clusters has been mentioned previously and the crux of the issue is that the algorithms that require the number of clusters as input usually are given the number by the "most significant knee" method "Knee Point Detection in BIC for Detecting the Number of Clusters — Request PDF", n.d. The integration of this method is a work in progress.

The dynamic pixel filtering refers to the self written filters as mentioned in 3.3 that use some input x, y or even more for their filter. These values should be identified using the characteristics of the image, however in this analysis the setting of these

values was fixed and common for all images. This means that images that are darker or lighter than the expected values were pruned unnecessarily. A method that finds the histogram of an image (see below) has been written, the integration would be to use the histogram to isolate the pixels that correspond to the particles before the Laplacian matrix is applied.

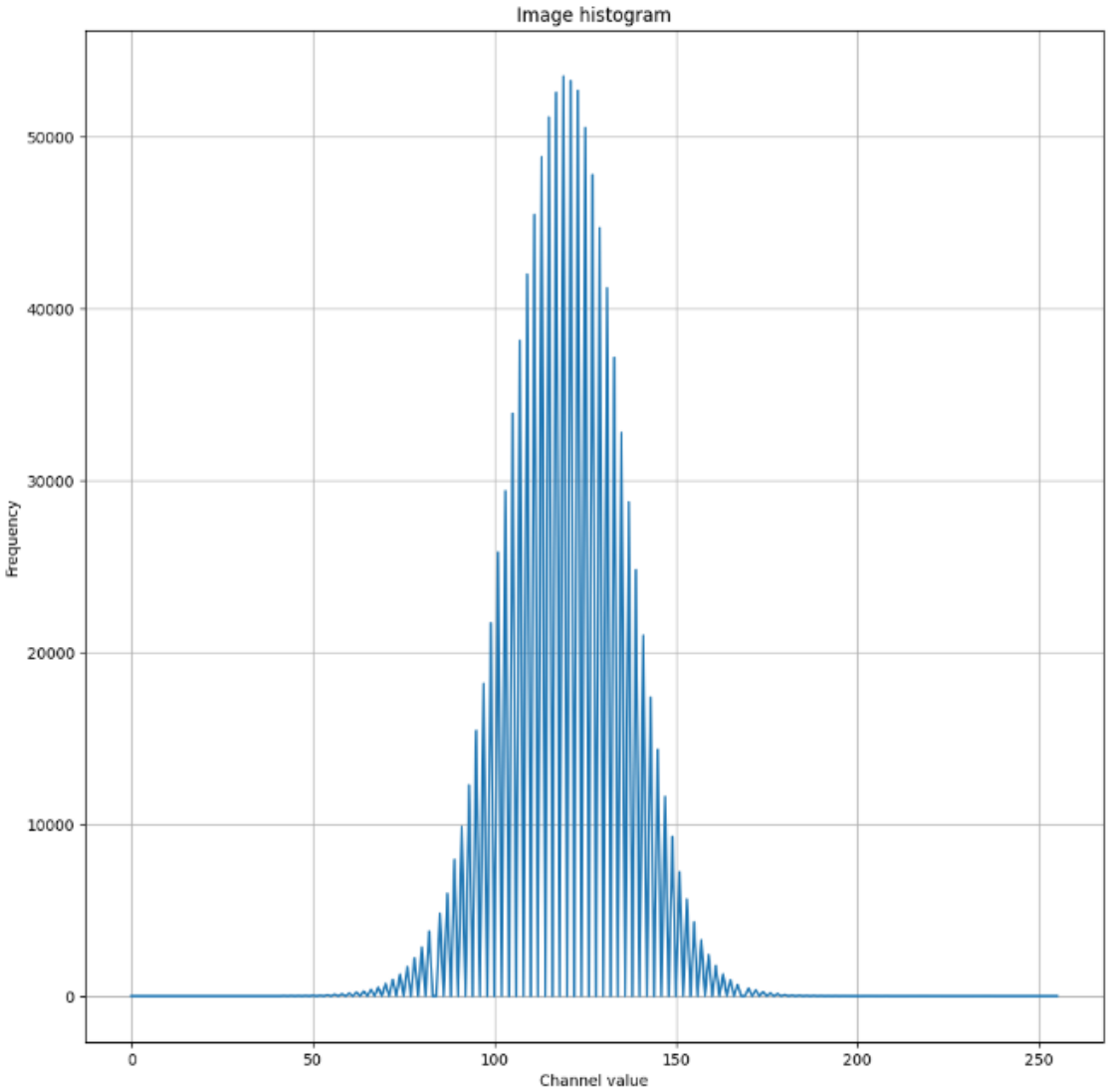


Figure 12: Example of the isolated histogram of an image

Each image has a different histogram yet we use the same range of pixels for all images.

Finally we can use the outputs of the clustering algorithms along with the knowledge of the conversion from units of pixels to units of lengths in the image to calculate the Euclidean distance of each point (pixel) that corresponds to a predicted particle. This would generate a gram matrix for each prediction and as such would be able to generate a putative structure from the consideration of all such

matrices using singular value decomposition "(PDF) Singular Value Decomposition (SVD)", n.d.

References

(PDF) singular value decomposition (SVD). (n.d.). In *ResearchGate*. Retrieved June 23, 2025, from https://www.researchgate.net/publication/331230334_Singular_Value_Decomposition_SVD

3.13.2 documentation. (n.d.). Retrieved April 5, 2025, from https://docs.python.org/3/

Al-Azzawi, A., Ouadou, A., Max, H., Duan, Y., Tanner, J. J., & Cheng, J. (2020). DeepCryoPicker: Fully automated deep neural network for single protein particle picking in cryo-EM. *BMC Bioinformatics*, *21*(1), 509. https://doi.org/10.1186/s12859-020-03809-7

Al-Azzawi, A., Ouadou, A., Tanner, J. J., & Cheng, J. (2019a). AutoCryoPicker: An unsupervised learning approach for fully automated single particle picking in cryo-EM images. *BMC Bioinformatics*, *20*(1), 326. https://doi.org/10.1186/s12859-019-2926-y

Al-Azzawi, A., Ouadou, A., Tanner, J. J., & Cheng, J. (2019b). A supervised clustering approach for fully automated single particle picking in cryo-EM. *Genes*, *10*(9), 666. https://doi.org/10.3390/genes10090666

Banerjee, A., Dutta, B., & Bandopadhyay, R. (2020). Development of cryo-electron microscopy for high-resolution structure determination of biomolecules in solution: 2017 nobel prize in chemistry. *National Academy Science Letters*, *43*(3), 303–305. https://doi.org/10.1007/s40009-019-00841-x

Brigham, E. O., & Morrow, R. E. (1967). The fast fourier transform. *IEEE Spectrum*, *4*(12), 63–70. https://doi.org/10.1109/MSPEC.1967.5217220

Chari, A., & Stark, H. (2023). Prospects and limitations of high-resolution single-particle cryo-electron microscopy [Publisher: Annual Reviews]. *Annual Review of Biophysics*, *52*, 391–411. https://doi.org/10.1146/annurev-biophys-111622-091300

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*(8), 790–799. https://doi.org/10.1109/34.400568

Chung, J. M., Durie, C. L., & Lee, J. (2022). Artificial intelligence in cryo-electron microscopy. *Life (Basel, Switzerland)*, *12*(8), 1267. https://doi.org/10.3390/life12081267

Dhakal, A., Gyawali, R., Wang, L., & Cheng, J. (2024). Artificial intelligence in cryo-EM protein particle picking: Recent advances and remaining challenges. *Briefings in Bioinformatics*, *26*(1), bbaf011. https://doi.org/10.1093/bib/bbaf011

Jaccard similarity - an overview — ScienceDirect topics. (n.d.). Retrieved June 23, 2025, from https://www.sciencedirect.com/topics/computer-science/jaccard-similarity

Kernighan, B. W., & Ritchie, D. M. (1988, April). *The c programming language* (2nd). Prentice Hall Professional Technical Reference.

Knee point detection in BIC for detecting the number of clusters — request PDF. (n.d.). *ResearchGate*. https:

//doi.org/10.1007/978-3-540-88458-3_60

Li, Z. (2022). Editorial: Methods in structural biology: Cryo-electron microscopy [Publisher: Frontiers]. *Frontiers in Molecular Biosciences*, *9*. https://doi.org/10.3389/fmolb.2022.1041386

Lin, H. (2025, March 21). *Moonpuck/chameleon_cluster* [original-date: 2018-11-03T02:59:12Z]. Retrieved June 23, 2025, from https://github.com/Moonpuck/chameleon_cluster

Matplotlib.axes — matplotlib 3.10.1 documentation. (n.d.). Retrieved April 28, 2025, from https://matplotlib.org/stable/api/axes_api.html

NumPy. (n.d.). Retrieved April 29, 2025, from https://numpy.org/

OpenCV: OpenCV modules. (n.d.). Retrieved April 29, 2025, from https://docs.opencv.org/4.x/index.html

Optuna - a hyperparameter optimization framework [Optuna]. (n.d.). Retrieved March 29, 2025, from https://optuna.org/

Ovall, J. S. (2016). The laplacian and mean and extreme values [Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.4169/a *The American Mathematical Monthly*, *123*(3), 287–291. https://doi.org/10.4169/amer.math.monthly.123.3.287

Scikit-learn: Machine learning in python — scikit-learn 1.6.1 documentation. (n.d.). Retrieved March 28, 2025, from https://scikit-learn.org/stable/

Xu, C., Zhan, X., & Xu, M. (2024, April 15). CryoMAE: Few-shot cryo-EM particle picking with masked autoencoders. https://doi.org/10.48550/arXiv.2404.10178

Zamanos, A., Koromilas, P., Bouritsas, G., Kastritis, P. L., & Panagakis, Y. (n.d.). Towards generalizable particle picking in cryo-EM images by leveraging masked AutoEncoders.