Spring 2018

Image Segmentation and Classification of Marine Organisms

Krishna Teja Vojjila
San Jose State University

Image Segmentation and Classification of Marine Organisms

A project report presented to

The Department of Computer Science

San Jose State University

In partial fulfillment of the requirements for the class

CS 298

By Krishna Teja Vojjila

May 2018

Abstract

To automate the arduous task of identifying and classifying images through their domain expertise, pioneers in the field of machine learning and computer vision invented many algorithms and pre-processing techniques. The process of classification is flexible with many user and domain specific alterations. These techniques are now being used to classify marine organisms to study and monitor their populations. Despite advancements in the field of programming languages and machine learning, image segmentation and classification for unlabeled data still needs improvement. The purpose of this project is to explore the various pre-processing techniques and classification algorithms that help cluster and classify images and hence choose the best parameters for identifying the various marine species present in an image.

Table of Contents

1. Introduction

Throughout millennia, humans have developed an amazing aptitude for pattern recognition. Once taught, identifying objects becomes easy, especially through vision. Even a child can differentiate a burger from a hotdog. But when it comes to processing millions of images with intricate details, all at the same time, computers make a lot more sense to use. They are not only faster than manual labor, but cheaper too thanks to some novel algorithms that have been developed to mimic the functioning of the human brain [5].

In the era of big data and the internet, identifying an image(classification) and what's in an image (detection and segmentation) is a routine task. From searching images of items, food or clothing to looking for places to visit, computer vision is being used every day by smartphones and computers.

One such application of computer vision is classifying the species of marine organisms present in an image. The Moss Landing Marine Laboratories in Moss Landing, California works towards the identification of marine invertebrate fauna. They study and collect organisms by submerging stacks of flat settlement plates called ARMS (Autonomous Reef Monitoring Structures)

Figure 1. ARMS being placed on a coral reef.

These structures are strategically placed in areas having sunshine and shelter. There is enough gap left between them to facilitate the natural growth of flora and fauna. These plates are left for a period of 6 months up to a year and then retrieved.

Figure 2. Photograph of a retrieved plate.

Analysis typically includes photography of each plate followed by visual identification of megafauna by expert taxonomists and DNA extraction for molecular identification.

Although ARMS plates have proven effective collection devices [20], current identification techniques have significant expense. Moreover, expert taxonomists are not always available, while molecular identification is time consuming. The photographs are generally archived without analysis. There is thus an opportunity to enhance current analysis practices by software analysis of ARMS photos.

This project proposes and implements a solution to automate the above task. It aims to identify the different species present on a plate by segmenting and classifying the different regions of the image into their respective classes. The exact classes would have to be later renamed according to their scientific names by domain experts. The process flow is outlined in the following diagram.

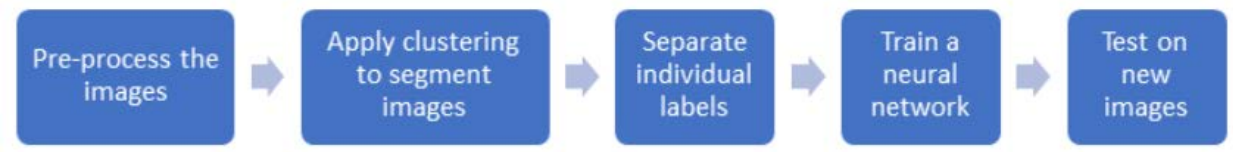


Figure 3. Process flow of the project.

2. Background

2.1 Supervised and Unsupervised learning

Before determining the available methods that should be used to solve the problem at hand, there is a greater need to understand the challenges and the reasoning behind choosing any methods. The presence of multiple overlapping organisms in an image combined with the fact that there are no labels for the images make it a difficult proposition. Any machine learning model with a predictive capability works better if it has labelled data to train the model on. Hence the first step is to generate labels for the images.

There are 2 broad types of learning techniques in the machine learning context - Supervised and Unsupervised Learning [13]. In supervised learning, the model is given data and explicitly told which class that data belongs to. In unsupervised learning, the model has no idea about the class information of the data. It tries to come up with its own classes based on some features of the data.

As the data provided is unlabeled, supervised learning is not feasible. Hence, the labels must be generated from the images themselves [10]. Not only that, there are multiple classes present in the image. Hence, the different classes need not be just identified, but also marked with some arbitrary feature (e.g. color) to differentiate them from one another. Such a task in machine learning is called as Image segmentation and is critical to this project.

There are many algorithms that perform unsupervised learning. One of them is clustering - the task of grouping similar set of objects in a given feature space. In the context of image classification, it has been traditionally used in segmenting medical [6] and satellite images [14]. Hence it makes sense to extend them to this project as well. The objects in this case are the intensity values of the different pixels in the image. Pixel clusters that represent an organism(class) should be like the other pixels of the same organism in shape, size, color or texture. Hence applying clustering analysis to the image would yield us different clusters for each organism. The beauty of this approach is that as each cluster would be intrinsically different from one another, the output would in fact be the different classes present in the image.

Clustering can be further distinguished into hard and soft methods [15]. Hard clustering treats an object such that it belongs to only one cluster or class. In soft or fuzzy clustering, each object is given a percentage or likelihood that it belongs to a cluster, hence it can belong to multiple clusters. Hard clustering is useful when the boundaries of different classes are fixed and linear whereas soft clustering is better in describing more complex decision boundaries. In this project, one algorithm of both the types is implemented - K-means of hard clustering and fuzzy c-means of soft clustering.

2.2 K-means algorithm

A computer does not view images the same way as humans do, it has no contextual information other than the labels the image has. For it, images are nothing but an array of RGB colored pixels with various intensity values.

K-means algorithm aims to partition n points into k clusters so as to minimize the inter cluster sum-of-squares(variance) [19]. The data points in this case are the pixel values of the image. The parameter 'k' must be passed beforehand and is integral to the performance of the algorithm. It starts with a 'k' randomly selected centroids and iterates between the two following steps: -

1. Assignment step

Each data point is assigned to the nearest centroid and its distance to the centroid is minimized. The metric used can differ, the most common one is the squared Euclidean distance.

$$\underset{c_i \in C}{\operatorname{argmin}}\ dist(c_i,\ x)^2$$

where c_i is the collection of centroids in the set C and x is the data point

2. Update step

In this step, each of the centroids are recomputed by taking the mean of all the data points in that centroids cluster.

$$c_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i$$

Where S_i is the set of data point assignments for each i^{th} cluster

One of the advantages of this algorithm is it will always converge. The result may be a local optimum, but multiple runs with randomized starting centroids can produce better results. The performance of the algorithm depends upon the value of k. Now, there is no set method to determine 'k', the user needs to run the program for a range of k values and see the results to determine the number of clusters. One of the metrics that is used to compare the results between different values of k is the mean distance calculated from the data points to their cluster centroid [7]. As we go on increasing k, the number of centroids increases and hence the mean distance will always decrease, up to the point of 0 when k equals the number of data points itself. Hence, this alone cannot be used. The mean distance is plotted alongside k and the point where the graph sharply falls, also known as the 'elbow point', can be used to determine the best value of k. It indicates that the value of k for which there is a sharp fall in distortion, i.e., mean distance from data points to their cluster centroid has decreased more than ever before, will not occur again that drastically. Hence there is no use of increasing k for more computations if the tradeoff is only going to be minute

improvements to distortion levels. Hence that value of k can be used as an optimum value as it explains the data with as little distortion as possible compared to the previous values of k.

2.3 Fuzzy c-means

Fuzzy c-means clustering is yet another algorithm for clustering. It assigns a membership to each data point depending on its distance to each cluster center. The nearer it is to the cluster center, the more its membership towards the same. The total sum of all memberships of a data point is one. The algorithm proceeds as follows [2]: -

1. Randomly select 'c' cluster centers.
2. Calculate the fuzzy membership u_{ij} of each point using

$$\mu_{ij} = 1 / \sum_{k=1}^{c} (d_{ij} / d_{ik})^{(2/m-1)}$$

3. Calculate the fuzzy centers v_j by

$$v_j = (\sum_{i=1}^{n} (\mu_{ij})^m x_i) / (\sum_{i=1}^{n} (\mu_{ij})^m), \forall j = 1, 2, ...c$$

4. Repeat step 2 and 3 till until J is minimized or $||U_{k+1} - U_k|| < \beta$

   Where β is a value between [0,1], a termination criterion

   K is the iteration step

   J is the objective function

   $U = (u_{ij})_{n*c}$ is the fuzzy membership matrix

However, having a lower value of β we can get better results, but there will be more iterations on the data. Also, the choice of distance metric can disregard underlying factors in the data and give unequal importance to them. E.g. Euclidean distance.

Fuzzy c-means clustering works particularly well with overlapped data set because the data points in the boundary can have common memberships between the overlapped classes hence better represent the data [12]. The two graphs below show the difference much clearer.

Figure 4. Comparison of hard and soft cluster memberships.

In the first graph of Figure 2, there are 2 clusters A and B, with members of A having membership 1 and B having 0. Thus, there is a clear delineation between the two clusters. In the second graph, the point in A marked by the dotted line is in between clusters A and B with a value of 0.3. Thus, that is the data points membership coefficient for class A [2].

The hardest part of any clustering algorithm is determining the number of clusters beforehand. The fuzzy partition coefficient or FPC is a metric used to determine the optimum number of clusters for a given data set. The FPC measure the amount of

'fuzziness' present in a solution, how close the solution is to the corresponding hard solution. This hard solution is formed by classifying each data point into the cluster which has the largest membership score. The formula is given below [2]

$$F(U) = \frac{1}{N} \sum_{k=1}^{K} \sum_{i=1}^{N} m_{ik}^2$$

Where m_{ik} is the membership of the i^{th} data point in the k^{th} cluster.

The value of FPC ranges from 0 to 1 hence the higher the value, the better the data is represented in the clusters. Thus, it will be used as a metric to determine the optimal number of clusters in fuzzy c-means clustering algorithm.

2.4 Convolutional neural networks

Once the images have been clustered and segmented into their various classes, the different labels for the images have been procured. Now, a supervised algorithm can learn the different labels and predict the same for new images. Artificial neural networks (hereafter abbreviated as ANN) are supervised learning models that are inspired from the human brain. The neurons in the human brain receive an input, fire up if the input signal is strong enough and pass its output to the next neuron. The brain learns by creating and destroying such connections and adjusting the strength of the signal.

ANNs operate similarly. In their case the input is a numerical array or matrix, be it images, text or plain numbers. It passes this input through multiple layers, where each layer comprises of neurons. Each neuron multiplies the input given to it with a weight associated with that neuron and then adds a bias to it, and then passes it to the next layer. The weights

and biases are present to ensure a similar input is received by the neuron across a range of acceptable values. Eventually, the last layer has an activation function that produces the final output. The activation function can be as simple as sigmoid function ($f(x) = (1 + e^{-x})^{-1}$) for binary classification(to classify 2 different labels) or as complex as ReLU($f(x) = \max(0, x)$) to classifying 'n' different labels. When the model is trained, the output for each iteration is compared to the true label. If it is incorrect, the weights and biases are updated accordingly, such that next time the same or similar input is received, the correct output is produced. A single neuron may look like the following [5]:-

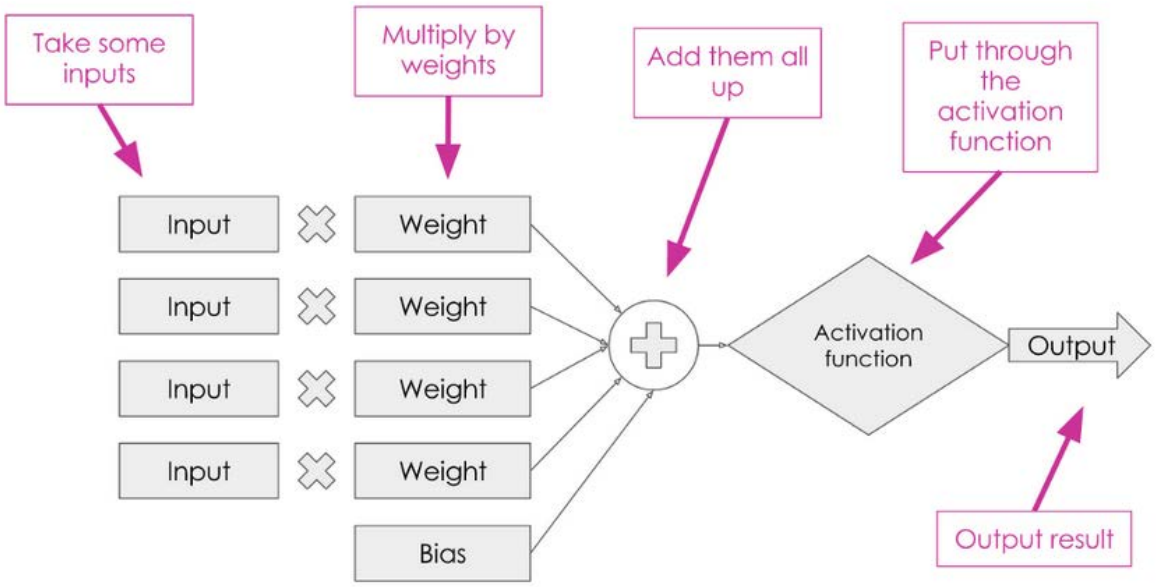


Figure 5. A sample neuron in an artificial neural network model.

Convolutional neural networks (hereafter abbreviated as CNN) are a type of artificial neural networks. Instead of taking a single input, a smaller array of numbers called the kernel filter is passed across the entire image multiple times. The values in the input image matrix are multiplied with the overlapping values in the kernel filter matrix and these

results are stored in a new array. This process is called convolution. Convolving images in such a manner produces many interesting effects. It can be used to detect edges, sharpen or blur the image, change the hue and saturation, etc. It is like applying filters to images in Photoshop.

Thus, for each layer, the model will learn a different feature. For the present project, it can learn the edges and lines in the first layer, different textures in the second layer, etc. till more complex organism shapes are learned in the final layers.

Another important concept of CNNs is pooling. Pooling is a down sampling technique which reduces the size of the input given to it while maintaining the relative information of the input data. One of the most common techniques is max pooling, which consists of outputting only the max value for every N X N grid of the input matrix. The intuition behind it is that the relative positioning of the pixels is more important than the absolute locations. Pooling layers are inserted in between the convolutional layers to reduce the number of parameters going through the layers. This decreases the amount of computational work and thus speeds up training time. It also controls overfitting.

A sample CNN neuron may look like this [5]:-

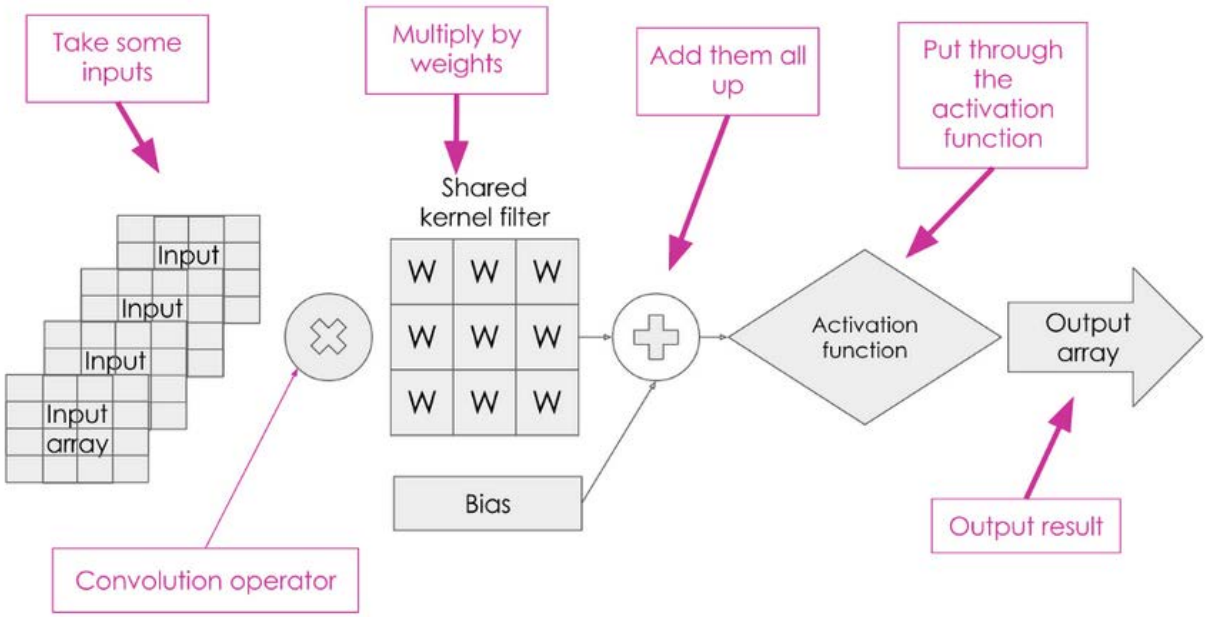


Figure 6. A sample neuron in a convolutional neural network model.

CNNs are very adept at identifying features in an image [1]. Unlike an ANN, where every single input or pixel is mapped to a single neuron, an area of the input image equal to the kernel filter size is mapped to each neuron. Thus, it operates on a set of pixels at once rather than a single pixel. It can learn more complex features by looking at the image at bigger scale when compared to an ANN, identifying textures, edges and lines.

CNNs enforce local connectivity patterns between adjacent neurons in a layer. This ensures that the learnt filters have a strong response to a local input pattern. Stacking layer after layer of such local patterns leads to formation of non-linear filters that become global, i.e., respond effectively to a larger pixel region.

CNNs feature different filters for each convolutional layer. However, each filter has shared weights and biases across that layer and thus form a feature map. Every neuron in that

layer shares these same weights and biases. This means that given a convolutional layer, all the neurons in that layer respond equally to the same feature. Hence no matter where the feature is present in the entire image, the layer would be able to catch it and output accordingly. This property is known as translational invariance. These two properties of CNN are what make them so adept at computer vision problems.

3. Method

3.1 Data description

6125 ARMS images from Monterey Bay, Hawaii and Guam were graciously provided by Prof. Jon Geller of Moss Landing Marine Laboratories. 200 images were used to train different models and a 100 were analyzed by testing on the resulting model.
All the images are of JPEG format and are divided into folders based on the geographical area they were taken from. They are further subdivided into collection sites A, B and C. Each image has an average size of 3 Megabytes with a resolution of 2848 X 4288, which is very high. Some of the images have the plate centered on it while some are out of focus, with different lighting as well. So, there is a need to standardize all the images before training the model.

All the programming was done in Python version 3.5 as it has extensive support for machine learning libraries and is the most popular and easy to learn language for the same.

All programs were run on a local machine having 16 Gigabytes of RAM, Intel core i7 processor and a 2 Gigabyte NVIDIA GeForce 940 MX graphics card. Hence there is no costly equipment or server cost associated with this project.

3.2 Pre-processing

A basic principle in machine learning is that the data needs to be prepared in such a way that it exposes the problem to the algorithms you intend to use. All algorithms make assumptions about the data; hence the data needs to be prepared and standardized keeping that in mind. Any flaw or irregularity in the data gets carried forward into the final results, called *training error*; and pre-processing is an essential step in avoiding training error.

3.2.1 Rescale data

Firstly, all the images were cropped and resized to a standard size of 2474 X 2455, having 1 Megabyte of size. This eliminates the unnecessary white background along with any shadows present in the image. All that information is not required by the model to predict the classes in the image. Moreover, due to the reduced size and resolution, the model will run faster as fewer calculations must be performed.

3.2.2 Histogram equalization

Consider an image that has its pixel values confined to higher ranges only. For e.g., brighter images will have their pixels confined to higher values than the darker ones. But a good image will have pixels that are from different parts of the image. Hence if a histogram of

pixel values is plotted, the “bad” image will have a compacted area with just the high value pixels on it. To make it a better image, the histogram needs to be stretched across its ends to accommodate all the different intensities present. Histogram equalization achieves better global contrast by spreading out the most common intensity values effectively [9]. It helps to better differentiate the foreground from the background if both are equally light or dark. In this case, histogram equalization is mainly used to separate the organisms that are spread out across the grey plate background and better reflect their true colors.

The key advantage of histogram equalization is its computational inexpensiveness. It is straightforward and is an invertible operator meaning the process can be reversed to get the original image back if needed. The only disadvantage is that it applies the histogram function equally to both the signal and the noise, i.e., it is indiscriminate. Thus, in some cases it can decrease the actual signal and increase the background noise.

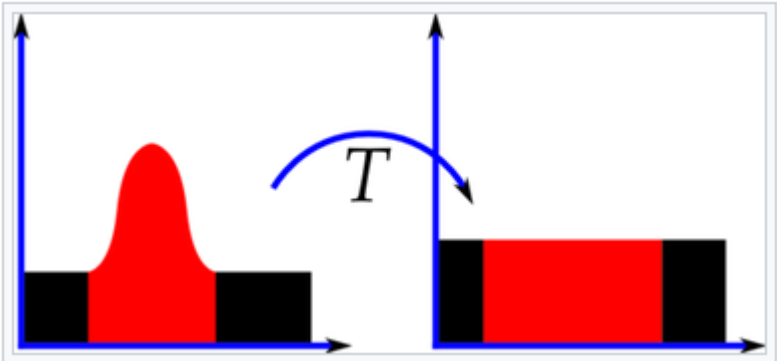


Figure 7. Histogram equalization.

3.2.3 Reshape data

When an image is read by the computer, it is converted into a 3-dimensional array of RGB values. But when it comes to giving this image as an input to the different models (k-means,

fuzzy c-means and neural network), the input is required in the form of a 1-dimensional array. The reason is that distance calculation and information representation is much easier and faster in 1D than 3D. There are far less parameters to worry about in 1 dimension and as all the values are relative, there is no loss of useful information as well. Once the processing is done, the 1-dimensional array is converted back into a 3-dimensional RGB array to display the colored image.

3.3 K-means clustering

To train the k-means clustering model, a folder named 'traindata' was created with 558 images from the Samoa collection site. Only a subset of the entire image data set is taken, as training and testing the entire dataset would take an enormous amount of time and resources. The images were read in one by one and the aforementioned pre-processing steps were performed on them, converting them from images into an array of pixels. The standard python library Scikit-learn was used to implement the k-means algorithm. Each image was fitted separately to the model having clusters ranging from 2 to 9 and the time taken for each cluster was noted down. The distortion values were also stored in a list to be analyzed later for finding the optimal value of 'k'. Then, the array of pixels was once again converted to an image by using the labels generated by k-means to color the pixel according to the corresponding label.

The output consisted of 8 images for the range of clusters 2 to 9 for each individual image. Each output image had a colored cluster that essentially segmented the image into different labels, in this case organisms.

As k-means is an unsupervised algorithm, it will treat every image as a new instance. In each image, it will cluster and the color the classes randomly. Hence, there is no continuity in the classes moving from one image to another. But, it is essential to maintain a constant class for each organism across the entire data set to train future models. Thus, a function was developed that outputs the cluster centers for each iteration and their corresponding pixel value. The steps performed in the function are listed below: -

1. Find the optimum number of clusters 'k' using the elbow method.
2. Output the average pixel values of all the 'k' cluster centers for all the images.
3. Create a list containing several colors in an order (light to dark).
4. Sort the pixel values in ascending order and assign a corresponding color for each cluster center.

Essentially, the label colors are hard coded using the average pixel values of the optimal 'k' cluster centers to bring consistency to the output images.

3.4 Fuzzy c-means clustering

The same data set used for k-means clustering was used here to compare the results effectively. The images went through similar pre-processing steps. The fuzzy c-means model from Skfuzzy library was used to fit the image across centers ranging 2 to 9. To calculate the optimum number of centers, the FPC value is outputted along with cluster center to graph out later.

The same function to hardcode colors to average cluster centers was reused here before segmenting and coloring the output images. The output image for each cluster center n was stored in a separate folder 'output' to be examined later.

The clustered images were ready to be sent forward to the final step of the classification. The k-means clustered images were chosen whose value of k was the best performing out of all the rest. The number of centers n for fuzzy c-means clustering was similarly chosen. Both the datasets were further sub-divided into their respective masks,i.e., 1 cluster per class of the image. Thus, k different masks per image were generated with just one particular cluster in each mask. It was achieved by writing the pixel values of each cluster to a blank image, one cluster at a time. This is done so that the neural network model can learn the different label independently and has clear labels for the same.

For e.g. consider the following image having 3 clusters. Thus k=3 or n=3 describes this data the best. The input to the CNN for this image would be the original image along with the 3 masks containing each individual points of the cluster one by one. Thus, the CNN would know these 3 labels are present in the input image and train to learn its features. The masks have the same resolution and size as the original image.

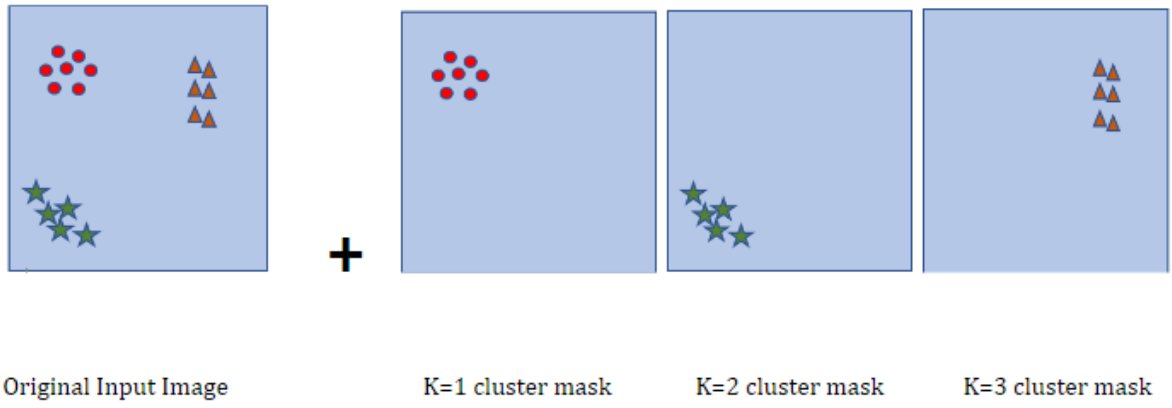


Figure 8. Image alongside its cluster masks.

3.5 CNN training

The input to the CNN were 2 sets of 100 images. These images were the results of k-means clustering and fuzzy c-means clustering respectively. Each image had multiple masks associated with it, which were its labels. The number of masks depends on the value of 'k' in the case of k-means clustering and the value of 'n' or number of centers in fuzzy c-means clustering. Each mask had the pixels of only one cluster. Each of the 2 sets of 100 images was divided into training and validation sets in the ratio 80:20.

The environment that was used to run the neural network was Keras [3]. Keras is a python library that is used to build and run neural network models. It provides an easy front end that helps to avoid the messy details of building neural networks from scratch. U-net is a neural network architecture that was originally created for segmenting bio-medical data such as MRI scans and blood vessel images [4]. So instead of having to design the individual layers of a CNN, a pre-built architecture like U-net can be used to train our model. U-net was already proven to perform very well on image classification problems and in this project, it is being extended to segment and identify marine organisms. This is known as transfer learning, where a pre-built model is being used for different machine learning tasks by training it on different data and labels.

The U-net model was imported directly to Keras. It had a total of 28 convolutional layers. The reason for choosing this architecture was its high performance even with low amounts of training data. As training neural networks takes a lot of time and resources, keeping the training set low while getting higher accuracy scores is a boon. The architecture of U-net is given below [4]: -

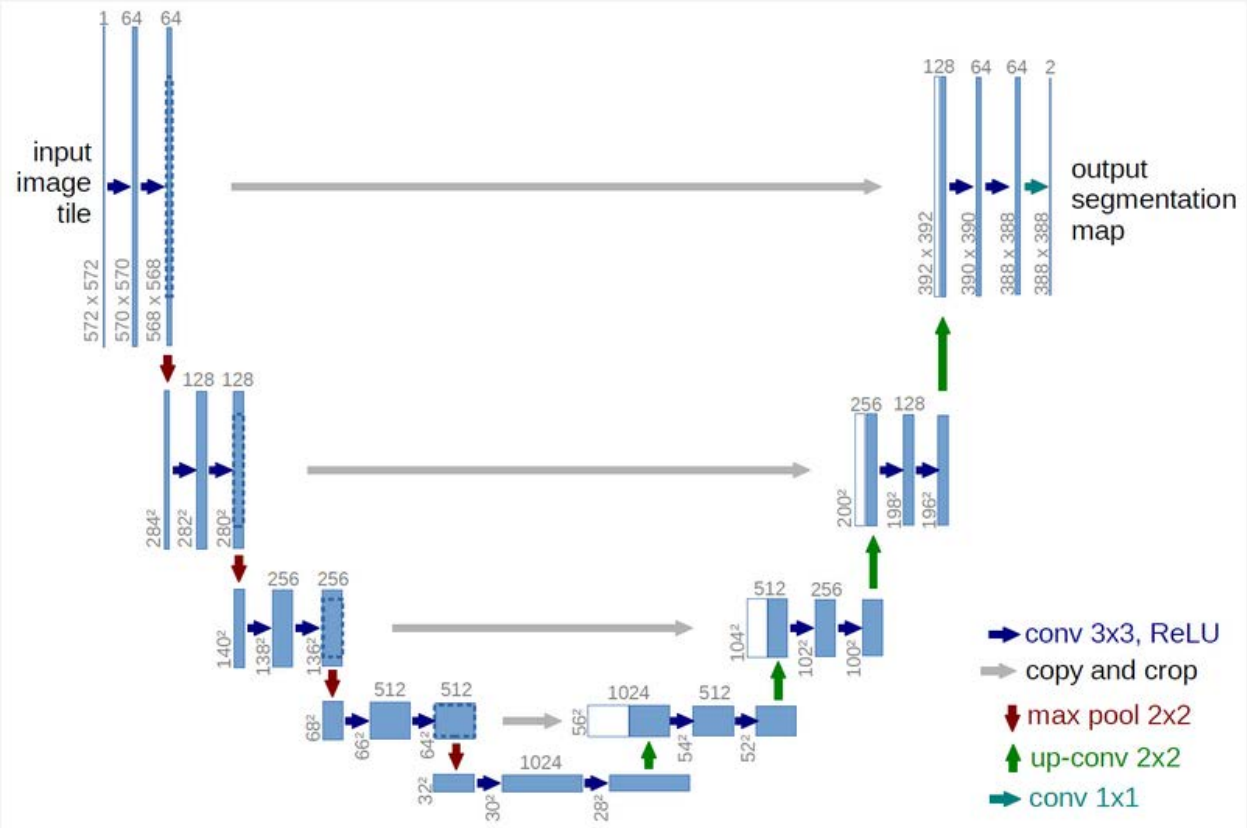


Figure 9. Architecture of U-net.

Now, the organisms present on the plate do not have a fixed size, shape or orientation. The same organisms may be miniscule in one image whereas be a majority in another. Hence the neural network model needed to be robust enough to identify it in all capacities possible. Thus, the input images were modified in different ways to generate more data. This is known as data augmentation. 3 functions were performed on the image.

1. Random rotation
2. Re-scaling/flipping
3. Hue/saturation change

If the original image was resized or flipped, the corresponding mask needed to be modified the same way to preserve the labels. If the hue and saturation of the input image were changed, the mask should not be changed, as the model is being built to identify only those labels, and they need to be fixed. Changing the hue and saturation is a way of telling the model that even though there might be slightly different shades of yellow in 2 different images, they are in fact of the same species. But if the orientation or size of the image was changed, the masks need to be changed as well as the model will insert the mask above the image to learn the features. Hence, they always need to be of the same resolution and orientation.

Next, the model was trained for both the training sets. The max number of epochs to train was set to 100 and the model loss at which to stop training was set to 0.20. The parameters were chosen thus to finish training in a reasonable amount of time. Ideally, when a test image is fed into the model, it will segment all the different organisms and output a separate image for each organism. Hence, when all the output images of the individual organisms are combined, it should give back the test image. The metric used to determine this is the dice coefficient [12]. It is defined as follows

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

Where X is the set of pixels that were predicted and Y is the ground truth. Basically, it determines how the predicted pixels match over the input image. Its value is between 0 and 1.

Loss is the summation of errors that the model makes on the training set. Maximizing the log likelihood (or minimizing the negative log-likelihood) means minimizing the distance between two probability distributions. Hence it is a measure of how similar our output is to the label. The lower the loss, the better the predictive capability of the model is with respect to its input. Ideally, every iteration of training should decrease the loss of the model till a certain threshold is reached.

Once the model was trained, the model loss and dice coefficient values are noted down. The best weights of the model were chosen and loaded into it. The model then worked on the validation or testing set of images. It generated an output for each of the identified labels in the image. The dice coefficient is once again used to measure the performance of the model.

4. Results

4.1 Pre-processing

The figure on the left shows the raw input image and the figure on the right shows the image after all the preprocessing techniques. The image on the right appeared standardized with even lighting and dimensions.

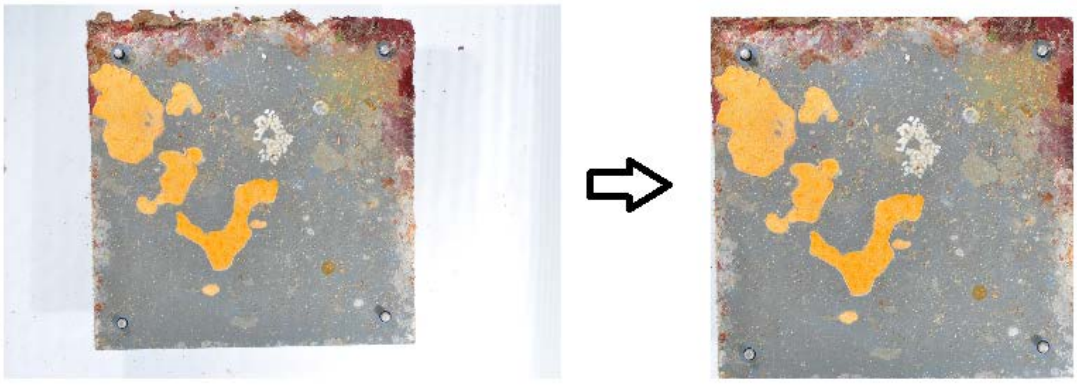


Figure 10. Image before and after pre-processing

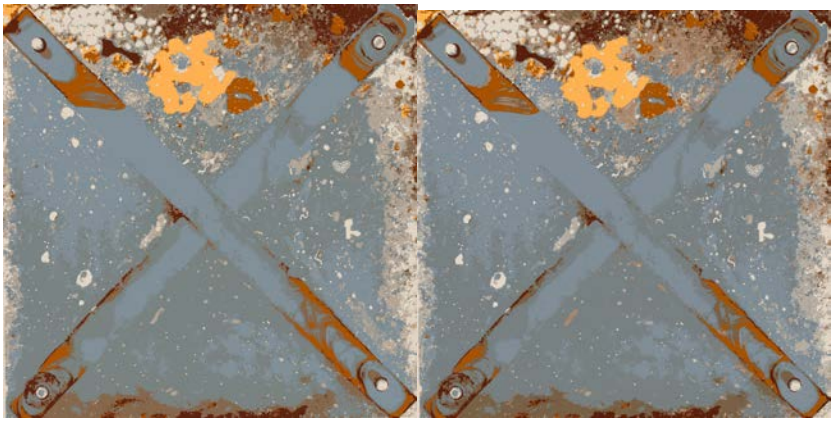
4.2 K-means

After performing k-means clustering on the images, 8 outputs per image were generated, with one output for every cluster k ranging from 2 to 9. The outputs for an example image look as follows: -

Figure 11. Input image for k-means clustering.

K=2 K=3 K=4

K = 5 K = 6 K = 7



K = 8 K = 9

Figure 12. Output images of k-means clustering for each k

As seen from the images, there is a gradual progress in how good the different organisms were being represented in the final segmented output. Looking at the output, it is obvious 2 clusters were not enough to represent all the different labels in the image, but the images with higher values of k look similar with not much difference. As the value of k increases, the k-means model has more centers to work with hence it can minutely segment the organisms in the image.

The graph below shows the average time take by the model to train k clustered images.

This is an important measure to evaluate the performance of the algorithm.

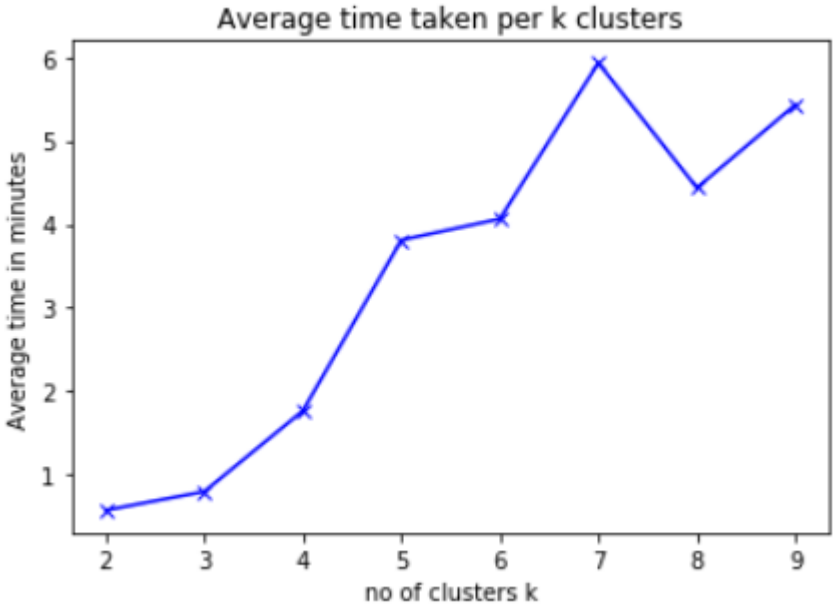


Figure 13. Average time taken to train k clusters.

Thus, when k=7 the model takes the highest amount of time to cluster the image.

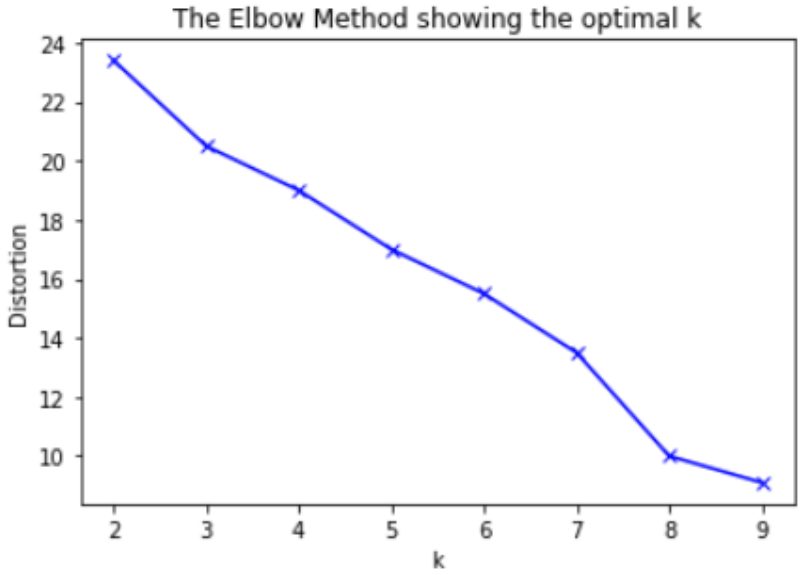


Figure 14. Elbow method to find the optimal value of k.

There is a sharp decrease in distortion when k=7. Hence that is the elbow point and the best approximate solution for finding k.

4.3 Fuzzy c-means

This model also outputted 8 images per image with the number of fuzzy centers ranging from 2 to 9. The average time taken to cluster for different values of n is graphed below: -

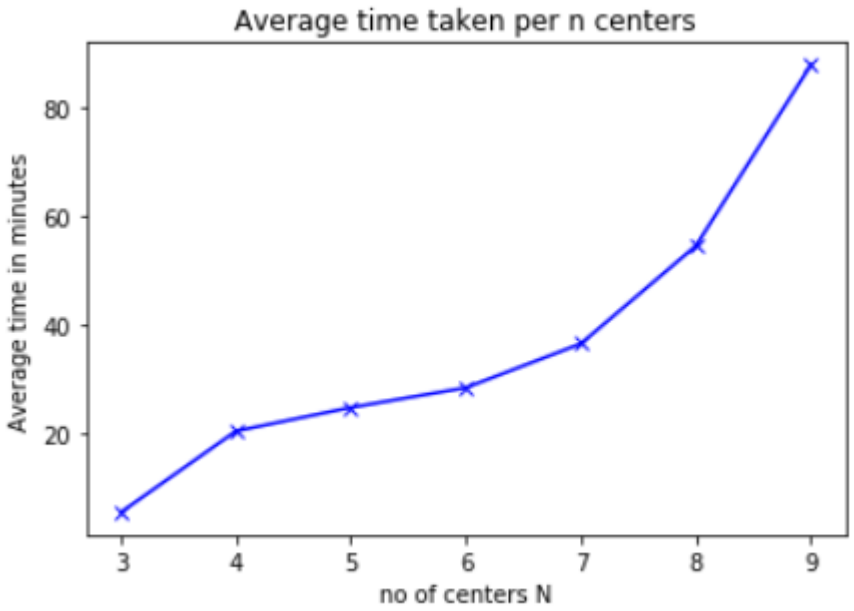


Figure 15. Average time taken to train N centers

The FPC graph is shown below which is used to estimate the best count of centers that can be used to segment the input image.

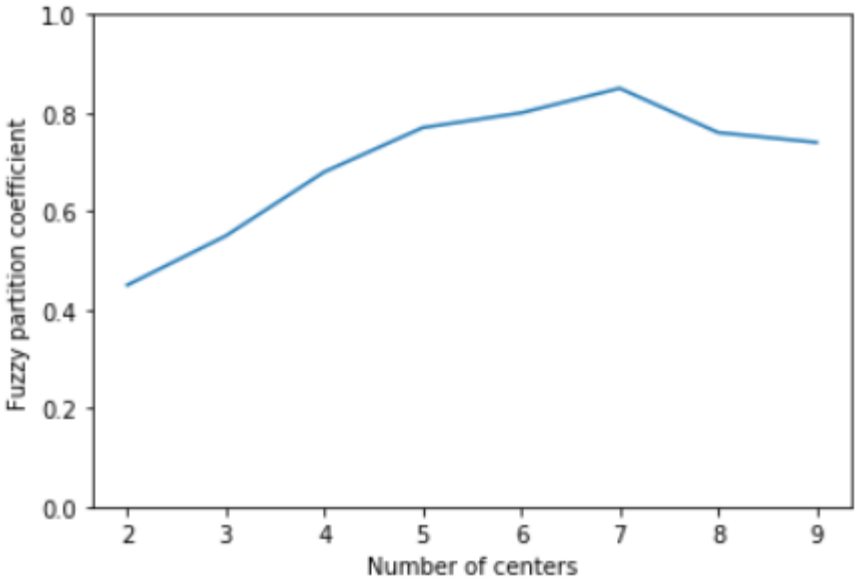


Figure 16. FPC for each N.

The value of FPC is observed to be highest at n = 7 where it is 0.85. Hence n=7 can be said to be the best value to represent all the classes in the image. The corresponding output of n=7 for the input image is as follows: -

Figure 17. Fuzzy c-means cluster output image for N=7.

All the similar organisms are of the same color whereas the background is nicely differentiated with a black color thanks to the histogram equalization which was done in pre-processing. The fuzzy c-means has taken advantage of the balanced-out intensity values to effectively separate the foreground from the background.

4.4 CNN output

For training the set of 80 images of k-means clustered images, the total time taken was 6 hours. For fuzzy c-means clustered images, the total time taken was 13 hours. The average dice coefficient and model loss on the validation set for both the dataset is graphed below: -

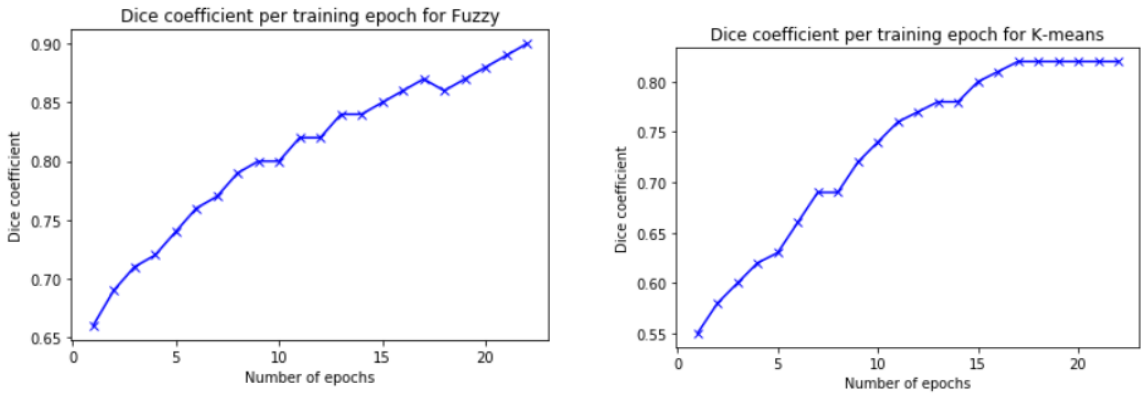


Figure 18. Comparing Dice coefficient per epoch for Fuzzy and k-means dataset on CNN.

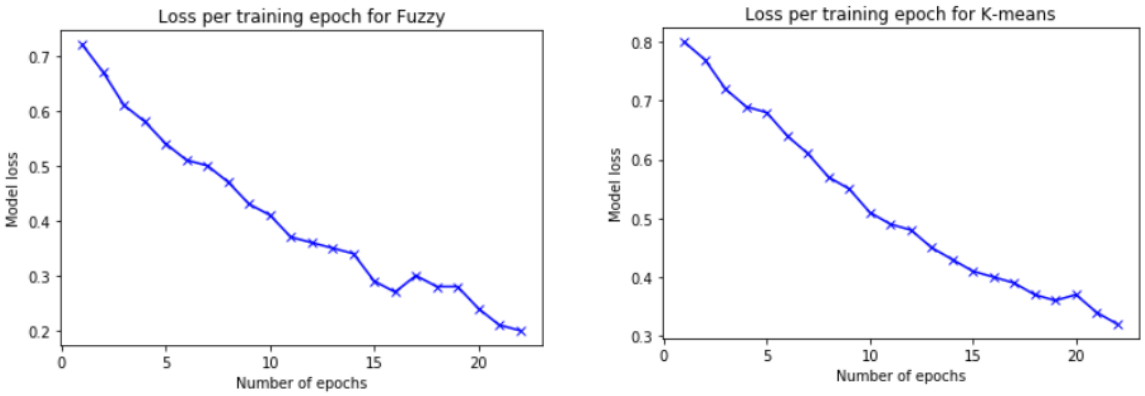


Figure 19. Comparing loss per epoch for Fuzzy and k-means dataset on CNN.

Below is the output generated by the CNN for a test image by the better performing fuzzy c-means clustered image model is as follows: -

Figure 20. Test image for CNN trained model.

Figure 21. Objects detected in cluster 1.

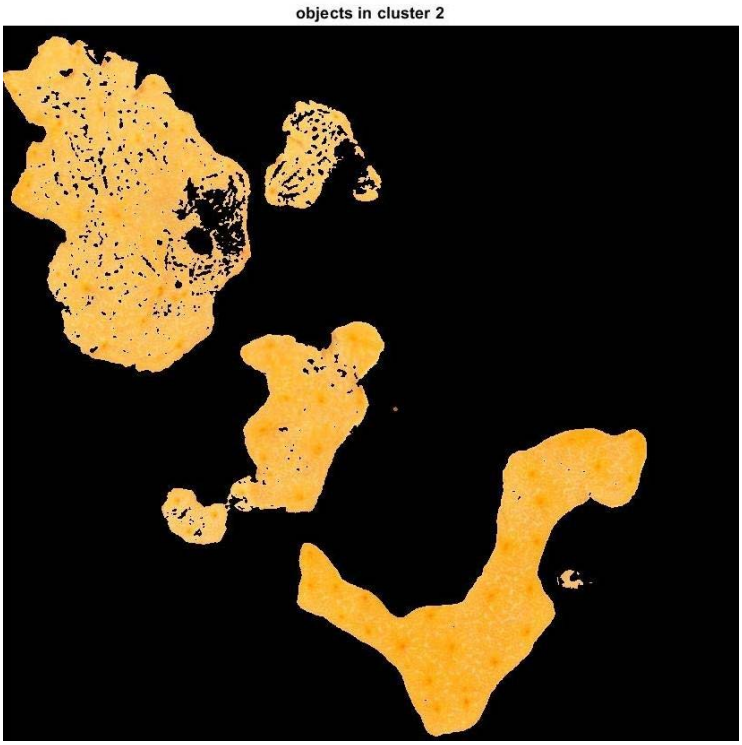


Figure 22. Objects detected in cluster 2.

Figure 23. Objects detected in cluster 3.

Hence, it is clearly observed that the model has identified the white pods in the right side of the image in the first cluster, the yellow organisms in the second cluster and the red colored organism covering the top left part of the plate. The scientific names of the different species need to be decided by domain experts. Judging by the output, the model was able to differentiate the different organisms quite well.

5. Discussion

Upon analyzing the various results, we observe a similar pattern for the 2 clustering algorithms. The time taken for clustering an image is directly proportional to the number of clusters chosen. Both the algorithms reach a certain threshold of number of clusters where

it is able to represent all the classes in the image to the maximum extent. In this experiment, the number turned out to be 7 in both cases.

It is also safe to conclude that Fuzzy c-means clustering performed better than the k-means clustering algorithm. It was able to better differentiate the different organisms and represented them on the clustered output images effectively. It is the reason the dice coefficient (0.90) was better on the fuzzy dataset rather than the k-means dataset (0.82). However, it is not without its tradeoffs. While the average clustering time for 8 clusters in k-means was 5 minutes, the same for fuzzy c-means was an enormous 87.5 minutes. The same could be said for training the CNN models where the fuzzy dataset took twice as long as the k-means dataset. The reason for this is the higher complexity of the fuzzy c-means algorithm. Instead of just one label per pixel like k-means, it has to calculate memberships of every pixel to all the cluster centers, and then take a mean. Thus, there are quite a lot of calculations per pixel [11].

But it is the same reason it is able to perform better than k-means. Whether a pixel belongs to a particular cluster is never crisply defined, hence there is more information available to the model to better judge the image. In k-means, because it is a hard-clustering technique, much of the information is lost due to hard coding the pixel labels to one cluster only.

Another observation made from running all the above experiments was that a lot of time and resources are required to cluster and train a model, especially when run on a local machine. In the future, an online server can be rented that contains state of the art Intel CPUs and NVIDIA GPUs to speed up the calculations and train even faster. GPUs have been shown to increase the performance of machine learning algorithms by up to 100 times because they are highly efficient at parallel processing. All the matrix multiplications, which

are integral to neural network models, can be performed simultaneously to gain drastic speedup in performance. More number of images can be clustered and trained resulting in additional accuracy and quality of segmented images. However, the budget of the project should also make it feasible to rent such servers.

6. Conclusion

Thus, a novel solution in the python programming language was implemented to classify the different marine organisms present in an image. The project showcased the importance of pre-processing techniques to prepare the data for an algorithm to properly work on it. It also demonstrated the effectiveness of clustering algorithms to perform unsupervised learning to create labels from an unlabeled image dataset. It compared k-means clustering and fuzzy c-means clustering techniques and found the latter to be more accurate than the former, but requiring more than 15X the time to perform the same task. Convolutional neural networks were also shown to be particularly adept at identifying different image features. The U-net CNN model, trained on fuzzy c-means clustered images, was able to predict and segment the various organisms with a dice coefficient of 0.90 which is impressive for a dataset as small as 100 images. The various output images containing the individual organisms from the original image will help researchers and scientists automate the identification process and move away from manual testing, thereby saving a lot of time, money and energy.

7. References

[1]"Convolutional neural network", En.wikipedia.org, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network#Distinguishing_features.
[2]"Fuzzy clustering", En.wikipedia.org, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Fuzzy_clustering.
[3]R. Relan, "A Non-Expert's Guide to Image Segmentation Using Deep Neural Nets", Medium, 2018. [Online]. Available: https://medium.com/@hanrelan/a-non-experts-guide-to-image-segmentation-using-deep-neural-nets-dda5022f6282.
[4]O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", Lecture Notes in Computer Science, pp. 234-241, 2015.
[5]R. Campbell, "Demystifying Deep Neural Nets – Manchester Futurists – Medium", Medium, 2018. [Online]. Available: https://medium.com/manchester-futurists/demystifying-deep-neural-nets-efb726eae941.
[6]C. Kirbas and F. Quek, "A review of vessel extraction techniques and algorithms", ACM Computing Surveys, vol. 36, no. 2, pp. 81-121, 2004.
[7]S. M.AqilBurney and H. Tariq, "K-Means Cluster Analysis for Image Segmentation", International Journal of Computer Applications, vol. 96, no. 4, pp. 1-8, 2014.
[8]A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks", Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017.
[9]"Histogram equalization", En.wikipedia.org, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Histogram_equalization. [Accessed: 09- May- 2018].
[10]L. Dong, L. He, M. Mao, G. Kong, X. Wu, Q. Zhang, X. Cao and E. Izquierdo, "CUNet: A Compact Unsupervised Network for Image Classification", IEEE Transactions on Multimedia, pp. 1-1, 2018.
[11]Z. Cebeci and F. Yildiz, "Comparison of K-Means and Fuzzy C-Means Algorithms on Different Cluster Structures", Journal of Agricultural Informatics, vol. 6, no. 3, 2015.
[12]M. Gupta, M. Shringirishi and D. Singh, "Implementation of Brain Tumor Segmentation in brain MR Images using K-Means Clustering and Fuzzy C-Means Algorithm", INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY, vol. 5, no. 1, pp. 54-59, 2006.
[13]A. Ahmad and S. Quegan, "Comparative analysis of supervised and unsupervised classification on multispectral data", Applied Mathematical Sciences, vol. 7, pp. 3681-3694, 2013.
[14]M. Nair and B. J.S., "Supervised Techniques and Approaches for Satellite Image Classification", International Journal of Computer Applications, vol. 134, no. 16, pp. 1-6, 2016.
[15]M. Waseem Khan, "A Survey: Image Segmentation Techniques", International Journal of Future Computer and Communication, pp. 89-93, 2014.

[16]R. Vieux, J. Benois-Pineau, J. Domenger and A. Braquelaire, "Segmentation-based multi-class semantic object detection", *Multimedia Tools and Applications*, vol. 60, no. 2, pp. 305-326, 2010.
[17]D. Marmanis, K. Schindler, J. Wegner, S. Galliani, M. Datcu and U. Stilla, "Classification with an edge: Improving semantic image segmentation with boundary detection", *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 135, pp. 158-172, 2018.
[18]K. Zou, S. Warfield, A. Bharatha, C. Tempany, M. Kaus, S. Haker, W. Wells, F. Jolesz and R. Kikinis, "Statistical validation of image segmentation quality based on a spatial overlap index1", *Academic Radiology*, vol. 11, no. 2, pp. 178-189, 2004.
[19]D. ZHENG and Q. WANG, "Selection algorithm for K-means initial clustering center", *Journal of Computer Applications*, vol. 32, no. 8, pp. 2186-2188, 2013.
[20]M. Williams, J. Ausubel, I. Poiner, S. Garcia, D. Baker, M. Clark, H. Mannix, K. Yarincik and P. Halpin, "Making Marine Life Count: A New Baseline for Policy", *PLoS Biology*, vol. 8, no. 10, p. e1000531, 2010.