Spring 2019

# PORIFERAL VISION

Saketh Saxena
*San Jose State University*

# PORIFERAL VISION

A project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfilment

Of the Requirements for the

Degree Master of Science

by

Saketh Saxena

May 2019

The Designated Project Committee Approves the Project Titled

Poriferal Vision

by

Saketh Saxena

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

San José State University

May 2019

Dr. Philip Heller Department of Computer Science

Dr. Leonard Wesley Department of Computer Science

Mr. Kevin Smith Department of Computer Science

## ABSTRACT

Poriferal Vision
by Saketh Saxena

Sponges provide nourishment as well as a habitat for various aquatic organisms. Anatomically, sponges are made up of soft tissue with a silica based exoskeleton which serves both as support and protection for the underlying tissue. The exoskeleton persists after the tissue decomposes, and microscopic parts of the exoskeleton break away to form spicules. Oceanographic studies have shown that the density of the sponge spicules is a good indicator of the sponge population in an area. This measure can be used to study sponge population dynamics over time. The spicule density is measured by imaging spicules from samples of water extracted from the oceans using an instrument called FlowCAM, which separates and photographs individual small items in a sample. It has a high processing rate, but is inefficient at computationally analyzing large numbers of photographs. Computer vision technologies, particularly deep learning using Artificial Neural Networks, and Support Vector Machines have shown to be effective in handling large scale image classification problems and are the de-facto standard in image recognition problems. Typically, these models require a large amount of data to learn the underlying distribution in datasets effectively and avoid model overfitting, which is currently a challenge to procure a vast dataset of images. To mitigate this challenge and achieve the overarching purpose of developing a high-performance classifier, we demonstrate various geometrical image transformation techniques to enhance the size of the dataset. We also show initial experimental results for training Generative Adversarial Networks for artificial synthesis of spicule images. Finally, we develop a Convolutional Neural Network and compare its performance against a Support Vector Machine for classifying images of sponge spicules training both the models on the original set of images and the newly generated set of images and achieve a test accuracy of 95% with a CNN trained on the newly generated images.



***Index terms* – Artificial neural networks (ANN), sponge spicules, bioinformatics, computer vision, deep convoluted neural networks (CNN), FlowCAM, generative adversarial networks (GAN), global silica biogeochemical cycle, image transformations**

## ACKNOWLEDGEMENTS

I would like to thank my project advisor Dr. Philip Heller, for his able guidance and encouragement throughout this project, and for giving me the flexibility to shape the final outcome of the project. I would also like to thank my committee members Dr. Leonard Wesley and Mr. Kevin Smith for their timely recommendations in developing this project and their constant support.

Finally, I would like to offer special thanks to Dr. Amanda Kahn of Monterey Bay Aquarium Research Institute (MBRAI) for providing the datasets that made this project possible.

# Table of Contents

# List of Figures

# List of Tables

# 1. Introduction

Sponges occur at all depths from intertidal zones to the continental margins and down to the abyssal zones ranging from 3,000 to 6,000 meters and hadal zones ranging from 6000 to 11,000 meters [1]. Sponges are primitive creatures which evolved about 500 million years ago [2]. There are about 15,000 extant species of sponges belonging to the phylum porifera which are further divided into three classes—Hexactinellida, Calcarea, Demospongiae [3]. Out of these the Class Hexactinellida commonly known as "glass sponges" occur in benthic regions of seas and oceans [3]. Anatomically, they are pale in color and are cup/basket shaped. They lack an epidermal covering and are made up of soft tissue supported and protected by an exoskeleton made up of spicules of silica which form a latticework. The exoskeleton composed of a grillwork of fused spicules persists after the underlying tissue decomposes [4].

Sponges provide nourishment as well as a habitat for various aquatic organisms. In the benthic zones, they can stabilize sediment and act as a substrate for larval recruits [5]. The Kahn [5] study describes in detail the impact of benthic grazing and carbon sequestration by deep-water glass sponge reefs in the oceanic ecosystem. Sponges can also filter organisms of microscopic scales of up to 1 to 50 μm from water [6]. The research studies by Chu [7] and Stryuf [8] show that along with diatoms and radiolarians, sponges are important local sinks of silicon and play a role in global silica biogeochemical cycle, inferring that sponges have a major role to play in the biological cycling of silica in the biosphere. Sponge populations are expected to rise with increase in ocean temperatures and decline of coral reefs. It is therefore important to study sponge population dynamics during past climate change events.

Studying glass sponges and their spicules is advantageous because the spicule particles persist over longer durations of time. Oceanographic studies of IODP sediment cores conducted by Moss Landing and Marine Biology Labs and Monterey Bay Aquarium Research Institute, have shown that the density of the sponge spicules is a good indicator or proxy for the sponge population in an area. The fluctuations in the quantity of spicules can be studied to monitor the changes in the taxonomy and abundance of sponges over time to predict the ecological conditions and changes thereof. It would also help in determining the factors which would allow for sustainable proliferation of sponge populations.

The spicule density is measured by imaging spicules from samples of water extracted from the oceans using an instrument called FlowCAM [9]. The resulting images are then manually inspected to identify and differentiate spicules from other particles such as diatoms and radiolarians which are henceforth referred to as "impurities." The impurities specifically radiolarians look very similar to spicules and may results in misclassification. For very large samples it takes a lot of hours of manual inspection. Another shortcoming of this approach is as the size of the image data increases storing and efficiently managing this data would become cumbersome since this process is not scalable.

Computer vision technologies, particularly deep learning using Artificial Neural Networks (ANNs), have shown to be effective in handling large scale image classification problems and are the de-facto standard in image recognition problems [10]. There are various types of neural networks but deep convolutional neural networks (ConvNets) and Generative adversarial networks (GANs) have shown the most promising results in object detection and classification [10], [11]. Although ANNs have been applied to some bioinformatics problems such as ecological modelling [12] and plant identification using vein morphology [13], their application in marine biology and

specifically to classify sponge spicules has never been attempted before. Most studies in regard to biological image data have been in Biomedical research [14]. In the later part of the 20th century, Support Vector Machines (SVMs) emerged as the state of the art for classification tasks due to their "high generalization performance" for high dimensional input without any prior knowledge [15]. We hypothesize that these powerful supervised learning algorithms would prove to be an effective and scalable means to solve the challenge of classifying spicule images.

Considering the need for a large-scale classification model, we propose to develop a scalable deep ConvNet for classifying sponges and other impurities and test its performance relative to a benchmark SVM model. With prescience we identify that a challenge we face in classifying sponge spicules is having a sparse dataset of images which could cause our classification models to overfit, in order to mitigate the problem of model overfitting we propose to increase our dataset by applying various image transformation techniques to the objects within the images and perform experiments on training the classification models using both the native set of images and the image set generated after applying various transformations henceforth described as "Transformed" image sets. We also describe our attempt to generate synthetic images using generative adversarial networks (GANs) to enhance the image set further [16]. The remainder of this report is divided into the following sections: section 2 describes the image gathering, training-test split and image transformations used, section 3 elaborates on the methods developed and used, section 4 enumerates our results, section 5 concludes the report and section 6 consists of a brief discussion about the project and a reflection of potential future work.

# 2. Image Gathering, Training-Test Split and Transformations

## 2.1. Image gathering: FlowCAM Imaging and Manual Identification

The data used in this project consists of FlowCAM images of radiolarians, diatoms and sponge spicules. FlowCAM is a proprietary machine primarily used to identify radiolarians and diatoms but can also be used to image spicules. Typically, the process comprises of collecting sediment core samples from the deep-sea and feeding these samples into FlowCAM, which then captures images of each microscopic particle one at a time [9]. These generated images are then manually segregated into two categories as depicted by Figure 1— (1) images of spicules and (2) images of diatoms, radiolarians and other microscopic particles. We consider images of spicules as the positive dataset and the other impurities as the negative dataset for the purposes of this project. The images generated are from sediment core samples extracted by the IODP Expedition 323, Sites U1340-1342 (Bering Sea) and provided by Dr. Amanda Kahn (MBRAI and Moss Landing and Marine Biology Labs). Figure 1 below shows three sample images of a spicule, diatom and radiolarians.

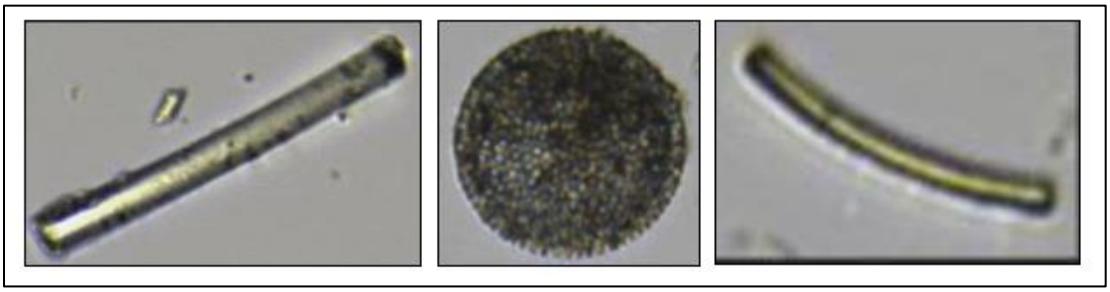

Figure 1 Sponge Spicule (left), Diatom (center) and Radiolarians (left)

The number of initial images for both positive and negative sets, along with the number of images manually selected for our experiments based on image quality are tabulated in Table I.

TABLE I. INITIAL DISTRIBUTION OF DATASET

| **Dataset** | **Type** | **Total Number** | **Number of selected images** |
|---|---|---|---|
| Positive image set | Spicules | 188 | 120 |
| Negative image set | Diatoms | 95 | 92 |
| | Radiolarians | 35 | 32 |
| Total Number | - | 318 | 244 |

## 2.2. Training-Test Split

A key challenge for the project which was identified early on was the scarcity of data for training and testing the classification models and GANs effectively. In order to ensure that the image transformations do not introduce bias in the final performance evaluation of the models, a set of positive and negative images was held out before applying any transformations. Another important consideration was to ensure that the test set was balanced and uniformly distributed and the number of images be significant enough to have meaningful performance metrics. We use about 25% of the positive and negative datasets to hold out for testing. The final number of images available for training and testing after pruning the test sets to balance them, are shown in Table II below.

TABLE II. TRAINING AND TEST SPLIT OF ORIGINAL DATASET

| **Dataset** | **Type** | **Training set** | **Test Set** |
|---|---|---|---|
| Positive image set | Spicules | 80 | 30 |
| Negative image set | Diatoms | 68 | 22 |
| | Radiolarians | 25 | 8 |
| Total Number | Total Number | 173 | 60 |

## 2.3. Image Transformations to Enhance the Training Dataset

In order to increase the number of images to mitigate the potential problem of model overfitting, we have applied 3 different image transformation techniques to the positive and negative image sets using OpenCV [17]. The image transformation techniques are described below:

i. Image Flipping

The original positive and negative images are taken as is and flipped around the vertical axis, horizontal axis and both the axes to produce 3 new images which increase the total number of images by 3 times.

ii. Perspective Transformations

Certain focal points within the image around the object are chosen and the object in focus is slightly shifted to the left side and top of the image to produce 6 perspective transformations. This transformation is applied on both the original positive and negative image sets.

iii. Rotational Transformations

The original images along with the 3 flipped images generated are rotated in 15-degree increments using the central locus of the image as the center.

The protocol followed to apply these transformations is depicted in Figure 2 below. In order to preserve the structure of the object in focus and get more realistic looking transformations the spicule or impurity in the image are detected using the chain approximation method and trace/draw highly accurate contours around the object [17]. A rectangular bounding box is drawn around the contours to isolate the image and preserve some of its background composition. A sample spicule image with a contour traced around the spicule and the bounding box in focus is shown in Figure 3 below.

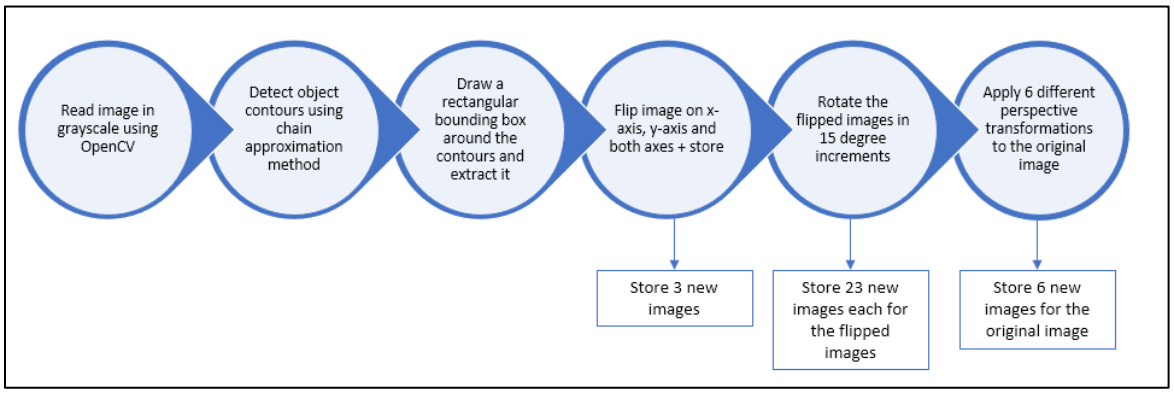


Figure 2 Image Transformation Workflow

Once the object has been isolated, the bounding box is flipped to generate 3 different axial orientations of the image which are then saved and perspective transformations are applied on these images to produce 6 transformed images, the flipped and original images are then rotated in 15 degree increments to generate 23 different orientations of the object. The image background is

then cleaned by removing noise and repainting any black edges or borders produced due to the rotations. Finally, the images are saved in .jpg format. The transformed image set is then checked for any blank images generated due to the perspective transformations by converting the images to gray scale and applying a gaussian filter on the image to blur the image if the majority of the pixels within the image are then black or white the image is removed from the dataset.

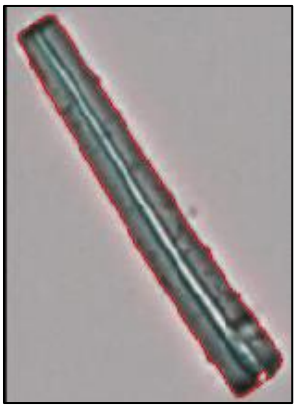

Figure 3 Bounding box of the spicule with a contour traced around its edges

As a final quality control step the images are manually checked for any images where the objects are cropped out and the ultimate transformed dataset is curated. As an attempt to generate more representations of the spicules we have also developed a method to develop a 3-D model of the image, but its application falls out of the scope of this project. The number of images generated and selected after applying all the transformations and manual curation are tabulated in Table III. Figure 4 below shows a sample set of images generated by applying the transformations described above.

TABLE III. NUMBER OF IMAGES GENERATED ON FLIPPING THE IMAGES, ROTATING THE IMAGES AND APPLYING PERSPECTIVE TRANSFORMATIONS

| **Dataset** | **Type** | **Original number** | **Flipped images generated** | **Rotated images generated** | **Perspective transformations** | **Total number of selected images** |
|---|---|---|---|---|---|---|
| Positive image set | Spicules | 80 | 240 | 5120 | 376 | 5816 |
| Negative image set | Diatoms | 68 | 204 | 4352 | 408 | 5032 |
| | Radiolarians | 25 | 75 | 1600 | 150 | 1850 |
| Total Number | - | 173 | 519 | 11072 | 934 | 12698 |

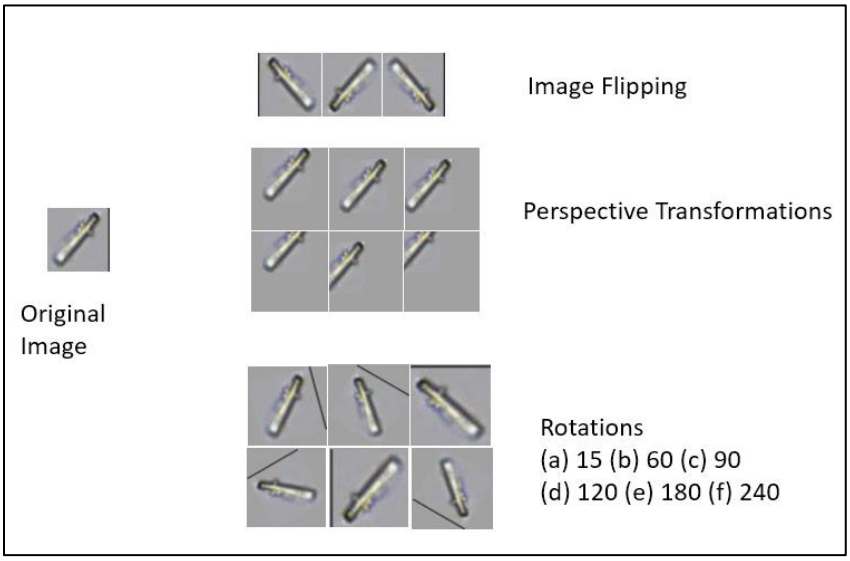


Figure 2 Sample results after applying transformations

## 2.4 Image Resizing

Due to the sparsity of the image dataset and high variation in the image resolutions ranging from resolutions as low as 26x64 to about 300x300, we resize all images to the same resolution. We adopt a naïve approach to choose a resolution to resize the image instead of using a standard 64x64 resolution as is common in many image classification algorithms. We compute the mean resolution of the training image set. Based on our computations we have chosen 138x78 as the resolution for the images.

# 3. Methods

## 3.1. Artificial Image Synthesis using GANs

Generative Adverserial Networks (GANs) was a novel framework proposed in 2014 for estimating generative neural network models using an adverserial process [16]. The fundamental idea behind the framework is to concurrently train two models namely— a generator "G" and a discriminator "D" such that G estimates the underlying distribution of the data and D estimates that a data samples originates from the original data as opposed to the outcome of G. Specifically for multilayer perceptron networks or neural networks, the framework uses backpropgation for simultaneous training of the generator and discriminator such that the generator is trained to maximize the probability of the discriminator returning a false positive result.

Goodfellow et al [16] have demonstrated that for arbitrary functions G and D there exists a unique solution where G learns the distribution of the training data and D results in about ½ for all input distribution. In literature this model is often elaborated using an analogy where the generator model is compared to a counterfieter and the discriminator as a forgery detector where

the generator tries to recreate a counterfiet item as closely as possible to fool the discriminator into believing the counterfiet item as real. Probabilistically, the GAN framework has been described as a two-player "min-max" game. Ever since the framework was proposed there has been a growing interest in the applications of GANs particularly in computer vision. Some of the applications include semantic image editing, image generation, text-to-image generation, style transfer and image classification [18]. We conceptualize a typical workflow while training a GAN in figure 5 below.

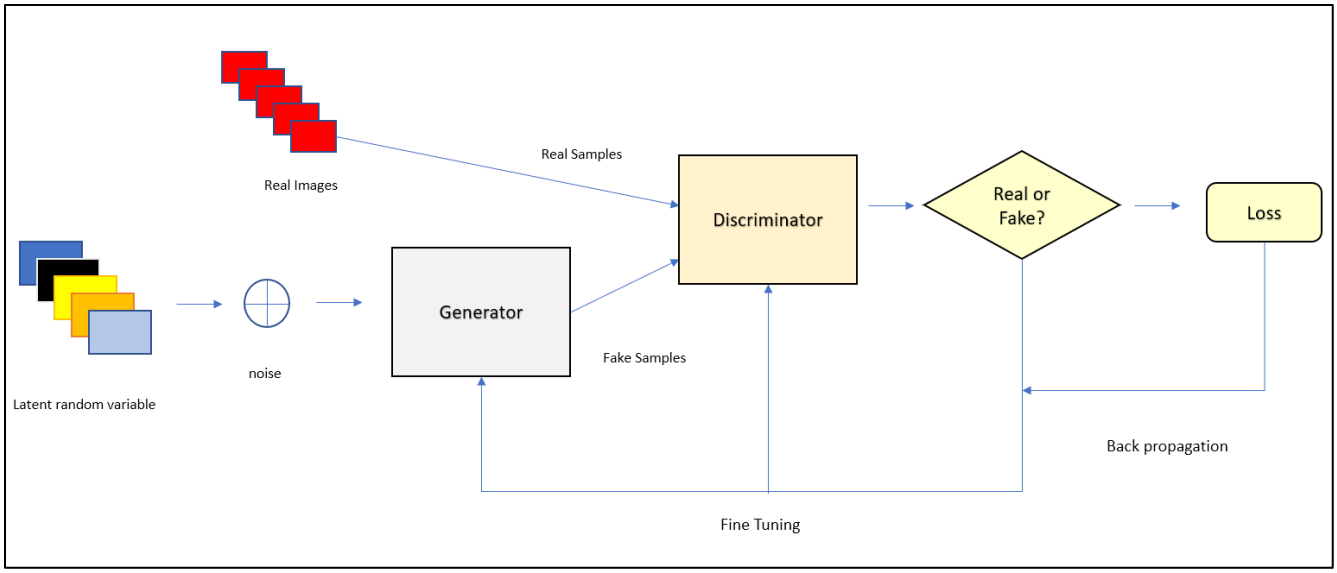


Figure 3 Workflow for Training a GAN based generative model

Many variants of GANs have been introduced in the recent years such as DCGAN [19] and Wasserstein GAN [20]. E. L Denton [21] introduced a deep generative image model which can produce high quality natural images soon after the release of the GAN frame work, the proposed model uses a laplacian pyramid of adverserial networks to generate extremely high quality images from sampled noise. Biological images generated using flouroscent mircoscopy typically have much simpler spatial and geometric structure than natural images but the variation in colors is an

indicator of important biological function , A. Osokin [22] demonstrated a novel approach to synthesize images of cell growth patterns as imaged from flouroscent microscopy loosely using the color channels for training a DCGAN, WGAN and WGAN-GP and proposed a Star based GAN architecture. Acknowledging the power of GANs in image synthesis, we have experimented with a "Vanilla" GAN and a DCGAN to generate artificial images with similar distribution as that of the training data. Both the Vanilla GAN and DCGAN are implemented using PyTorch which is a Python based open source deep learning platform [23].

### 3.1.1. Vanilla GAN

The Vanilla GAN we have developed is based on the original multi-layer perceptron network-based framework as proposed by Goodfellow et al [16]. We visualize the architecture of the Generator and the Discriminator Networks in Figure 20 in the appendix section using torchviz which is a python package to generate DOT format graphs for perceptron networks in PyTorch and render the graphs as JPEG files using graphviz [24].

The generator network consists of 3 hidden layers and an output layer, all the layers are feed-forward and apply linear transformations to the input samples. We use Leaky rectified linear unit activation function (LeakyRelu) in the layers with the alpha value as 0.2, which allows a small non-zero gradient even when the node is inactive [25]. The first layer inputs a tensor from a random noise sample of size 138 and returns a 256-channel tensor as input for the second layer. The succeeding layers apply the activation function and convert the samples into 512, 1024 and finally to an output size of 32,292. The output layer applies the element wise Tanh function to the generated sample. The discriminator network also has 3 linear, feed-forward hidden layers and an output layer. All the layers use a LeakyRelu activation function with the alpha value as 0.2 and

they contain a dropout function for regularization at each layer [26]. The first hidden layer takes as input the real or fake image samples with 32,393 features and the transformations at each layer are shown in Figure 20 in the appendix section. The final output layer applies the element-wise sigmoid function.

For the global parameters of the GAN we use the binary cross entropy loss function as the loss function and the "Adam" optimizer [27].

### 3.1.2. DCGAN

DCGANs were conceptualized as a class of unsupervised CNNs for learning image data representations using the adversarial net framework [19]. They had adopted three primary developments in CNN architectures for developing more stable DCGANs— (1) Replacing spatial pooling functions/layers with "strided convolutions" [28]; (2) Eliminating fully connected hidden layers [29]; (3) Mitigate shortcomings in training due to inferior initialization approaches by using batch normalization and enable gradient flow in deep multi-layer perceptron models [30].

Our model follows the architecture introduced by A. Radford, L. Metz and S. Chintala [19]. The generator network consists of a linear feed-forward layer followed by 4 2-d transposed convolutional layer, the first three convolutional layer applies batch normalization to the input sample and a ReLu activation function with an initial alpha value of 0.2 and the last convolutional layer does not apply either of the functions to allow for a deeper architecture. The output layer applies element wise TanH function. The discriminator network consists of 4 2-d convolutional layers and apply batch normalization at each layer followed by the LeakyRelu activation function with the standard alpha value of 0.2. The output layer of the discriminator network applies the element-wise sigmoid function to generate output. We continue using binary cross entropy as our

loss function and the Adam optimizer in our experiments with the DCGAN. A general schematic of the DCGAN we implementation is shown in Figure 6 below. For the entire architecture and parameters graph of the DCGAN we refer interested readers to Figures 21 and 22 in the appendix section.

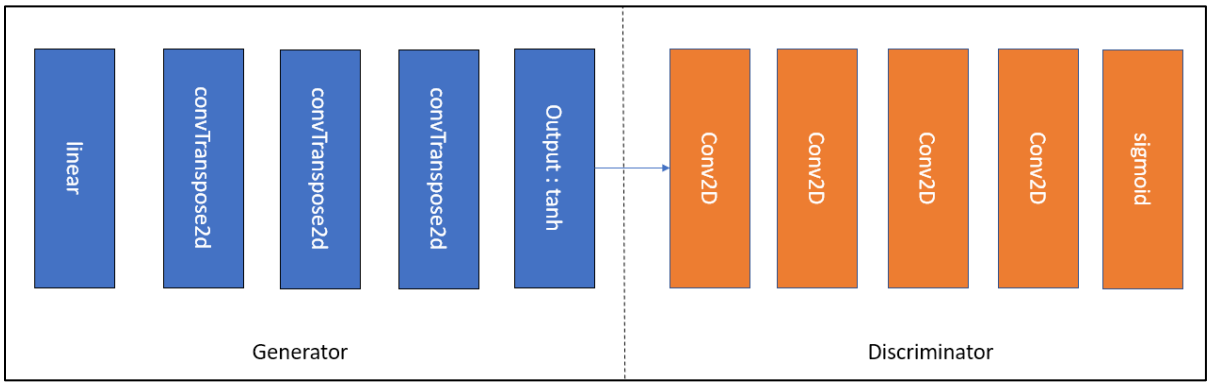


Figure 4 Schematic Representation of DCGAN

### 3.1.3. Image Preprocessing for GANs

Training GANs requires a huge amount of data and significant hardware resources to ease this process we perform the following transformations to the input dataset:

i. Compute mean resolution of the training data
ii. Resize the images to the computed mean
iii. Transform the images to grayscale and read in a single pixel channel
iv. Convert raw images to tensors using PyTorch transforms
v. Normalize the images

This protocol was used for both Vanilla GAN and DCGAN. However, for DCGAN we omit steps i-ii and resize the images to a fixed resolution of 64x64 and read in all 3 channels which is due to architecture constraints to reduce training time and the ability of DCGANs to handle

multi-channel input respectively. We use PyTorch transforms to perform the preprocessing steps for GANs [23].

### 3.1.4. Protocol for Training

We follow the same protocol for training both the Vanilla GAN (VG) and the DCGAN (DG). After the performing the image processing steps we train the GANs on two datasets, i.e., the smaller original dataset and the larger transformed dataset to generate two separate models each for VG and DG. We log the time taken for each epoch along with the generator and discriminator loss and display 6 generated images every 100 epochs and look for convergence of the model to stop training. The model's state parameters are saved as training checkpoints for every 100 epochs. We train VG for about 200 epochs with a batch size of about 500 for the larger transformed dataset and for about 200 epochs with a batch size of about 20 for the smaller dataset.

### 3.1.5. Performance Evaluation

In our experiments we test the training performance and convergence of the GAN for both the models and tabulate the generator and discriminator loss for various levels of training.

## 3.2. Image Classification Models

Image classification is a primary task in computer vision and machine learning. In order to develop a scalable machine learning model to classify spicules vs other impurities we train, and test two popular models commonly used for image classification tasks—SVMs and Deep ConvNets.

We note three primary reasons for selecting a SVM and a ConvNet model for the image classification task as opposed to simpler classification method—(1) With foresight, although the current dataset sizes are limited we anticipate that the number of images to classify would increase significantly by some orders of magnitude with further expeditions and using native image processing based classification methods such as template matching or feature matching using OpenCV would be accurate but take a significant amount of time for larger datasets; (2) As shown in figure 1, the images of spicules and diatoms are distinctive but the negative set also contains a substantial number of radiolarians which are visually very similar to images of spicules which would affect the accuracy of native approaches greatly and finding a good distance metric, template and feature set for the entire larger dataset would be quite cumbersome primarily due to the expected variability and (3) Finally, in conjunction with the above reasons and the overarching aim of the project to be able to study the variation in taxonomy and abundance of spicules, we aim to establish our ConvNet as benchmark binary classifiers for this task which can be further extended to perform multiclass classification and classify the spicules and impurities based on their species.

### 3.2.1. Support Vector Machine as a Benchmark

Support vector machines first came into prominence in 1995 and have seen immense amount of research and application particularly for binary classification tasks [31]. We use the Support Vector Classifier implementation in SciKit-Learn [32] with a linear kernel to train a support vector machine model which we use as a benchmark model to validate the performance of our CNN model.

#### 3.2.1.1. Principal Component Analysis of Images

As a byproduct of resizing our training sets based on a mean resolution, the number of features greatly increase while this would not be a problem for training the SVM on a smaller dataset, the training time for the SVM increased greatly once the size of the dataset was increased by adding the transformed images. We also conjecture that the performance of the SVM classifier could be affected by the resizing of the images. R. Sahak et al [33] used an approach combining SVM with PCA to reduce the feature size and achieved a high classification accuracy. To have a strong benchmark SVM classifier we perform principal component analysis on the image data to extract the most representative features from the images, and experimentally set the number of components to reduce the image dimensionality to a lower dimensional space.

#### 3.2.1.2. Parameter Optimization and Cross-Fold Validation

We use sklearn's GridSeachCV to optimize the c and gamma parameters while training the SVM along with 3-fold cross validation on a linear SVM kernel and return the best parameters and model which performed the best [32].

### 3.2.2. Deep Convolutional Neural Networks

Deep Convolutional Neural Networks (CNNs) can learn high dimensional image mappings for large samples of image data and have become an obvious choice for image recognition and classification tasks [34]. One of the main reasons for the revival of CNNs in the last decade or so was due to the availability of massive collection of datasets with comprehensive annotations and data distribution which allows for learning more generalized models such as ImageNet [11], [35].

Advancements in graphics processor units have also been a core driver in catalyzing the research in CNNs and achieving state-of the art results on image processing and detection tasks [36].

Specifically, for image classification tasks CNNs achieve better accuracy as compared to other machine learning models. AlexNet was a land-mark CNN architecture which had significantly higher classification accuracy as compared to other image classification methods and laid the foundation for further research and development of various CNN architectures customizable for a variety of domains [11], [36].

The deep CNN we developed comprises of two 2-D convolutional layers, followed by a dense layer with a relu activation function and an output layer with a sigmoid activation function [25]. The model implementation and all the visualization were done using TensorFlow [37]. The detailed architecture of our CNN model is shown in Figure 23 in the appendix.

### 3.2.3. Image Preprocessing for Classification

We perform the following image transformations to get a uniform resolution of all the images to train and test our classification models:

i. Read in the entire training set and convert the images to grayscale
ii. Compute the mean resolution of the images in the training set
iii. Resize the images to the new computed mean
iv. Normalize the image data
v. Assign class labels to the data where we use the standard notation of '1' for positive images and '0' for negative images
vi. Pickle out the dataset

All the above transformations have been implemented using OpenCV in Python [17].

### 3.2.4. Training and Experiments

We train two sets of classification models, i.e., one each of the SVM and CNN on the original dataset and the on the larger transformed dataset. For the SVM we especially perform a PCA step as described in section 3.2.1.2., and reduce the dimensionality of the feature space. In order to choose the number of components to reduce the input space to by fitting the principal component analysis algorithm to the dataset and evaluating the "Explained Variance %" ratio. We would ideally like the value of the Variance percentage high as we hypothesize that it would return a more representative feature set of the entire image set. We consider the entire dataset including positives and negatives while performing principal component analysis.

Once we choose a set of features, we perform PCA on our entire dataset and train the SVM model accordingly. We use scikit-learn's features to generate the model's evaluation result. While PCA helps in reducing training time greatly, we conjecture that the difference would not be statistically significant to learn the representative data distribution by the CNN. Therefore, we supply the resized version of the original and transformed images. We compute the training accuracy and training loss. Finally, we test out both the SVM and the ConvNet against the held-out test set.

### 3.2.5. Performance Evaluation

We evaluate various standard performance metrics used in image classification tasks for testing our classification models such as test accuracy, F-Score, precision and recall.

# 4. Results

## 4.1. Histogram Analysis of Images Generated from Transformations

We overcome the challenge of scarcity of data for training by enhancing the size of the dataset by applying various image transformations, it is imperative to ensure that the transformations do not add any bias in the data distribution. To validate our approach of applying transformations we perform histogram analysis on the original and transformed images. In figure 7, below we show the histogram plots for a single sample image to show the variation in the pixel intensity distribution between the original image and the mean of the transformed set of images including— (a) the histogram of the original sample image (b) the mean histogram of the images generated by flipping the image across both vertical and horizontal axes, (c) the mean histogram of the images generated by applying rotations and (d) the mean histogram of the images generated by applying perspective transformations on the sample image.

We also analyse the image histogram on the entire original image set and the entire set of images generated from transformations for both positive and negative datasets. The histograms thus generated are depicted in figure 8—Figure 8(a) shows the mean histogram for the entire positive training set comprising of the original images and 8(b) shows the mean histogram for the entire positive training set comprising of both the original images and transformed images. Similarly, figure 8 (c) shows the mean histogram of the entire negative training set comprising of only the original images and 8 (d) shows the mean histogram for the transformed training set and we observe that in both the cases the transformations produce the same peak values and distribution. In both figures 7 and 8 we normalize the frequencies of the distributions to account for images of different resolutions and sizes and allow for meaningful interpretation of the pixel intensity distributions.

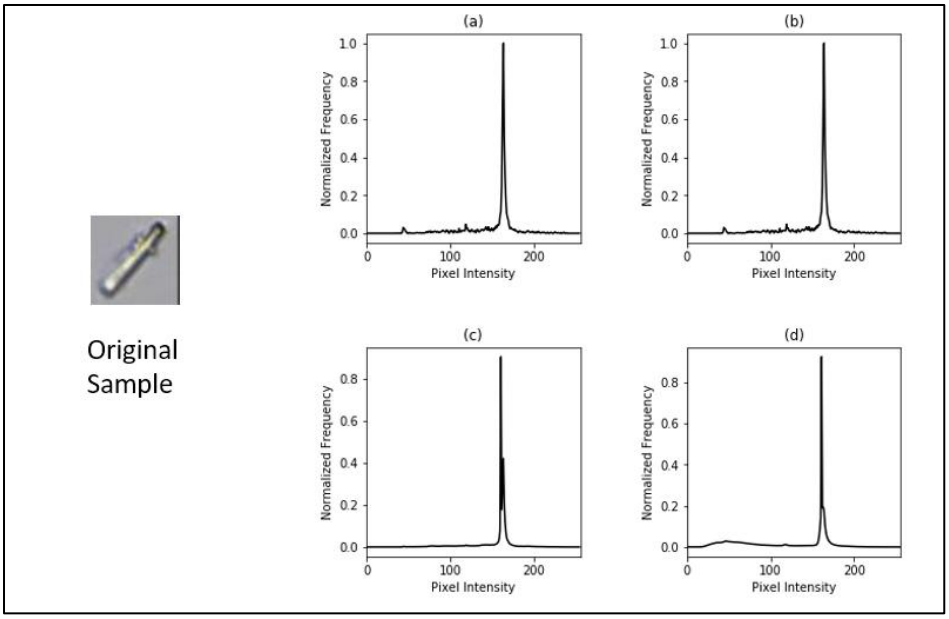


Figure 5 Pixel Intensity of histograms for the sample image—(a) Histogram of Original Sample Image in Grayscale; (b) Mean Histogram of Images generated by flipping the sample Image; (c) Mean Histogram of images generated on rotating the sample; (d) Mean Histogram of images generated on applying perspective transformations to the sample image

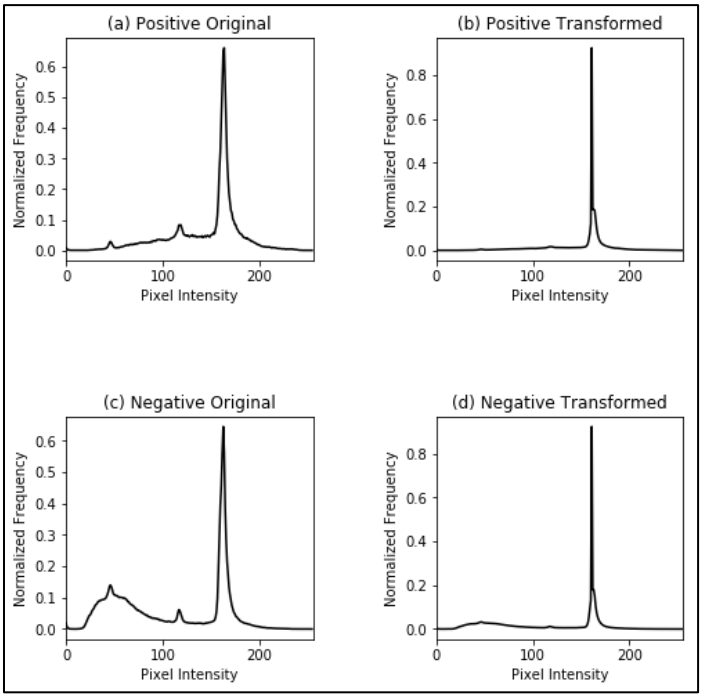


Figure 6 (a) Mean Histogram of Positive Training set; (b) Mean Histogram of Positive Training set with transformed images; (c) Mean Histogram of original negative Image sets; (d) Mean histogram of transformed negative training set

## 4.2. Training results for GANs

We present some initial results of our experiments with training GANs for image synthesis using only the spicule images of original dataset, we have run some experiments on the transformed dataset as well, but their results were largely inconclusive and insufficient and thus we leave them out of the scope for this report. In figure 9, we show the G-error and D-error which is representative of the loss during training for training vanilla GAN, the errors spike quite a bit initially as expected but soon converge at around 100 epochs but the model goes into mode collapse where the generator fixates on a certain distribution within the image which the discriminator always fail to identify as fake data.

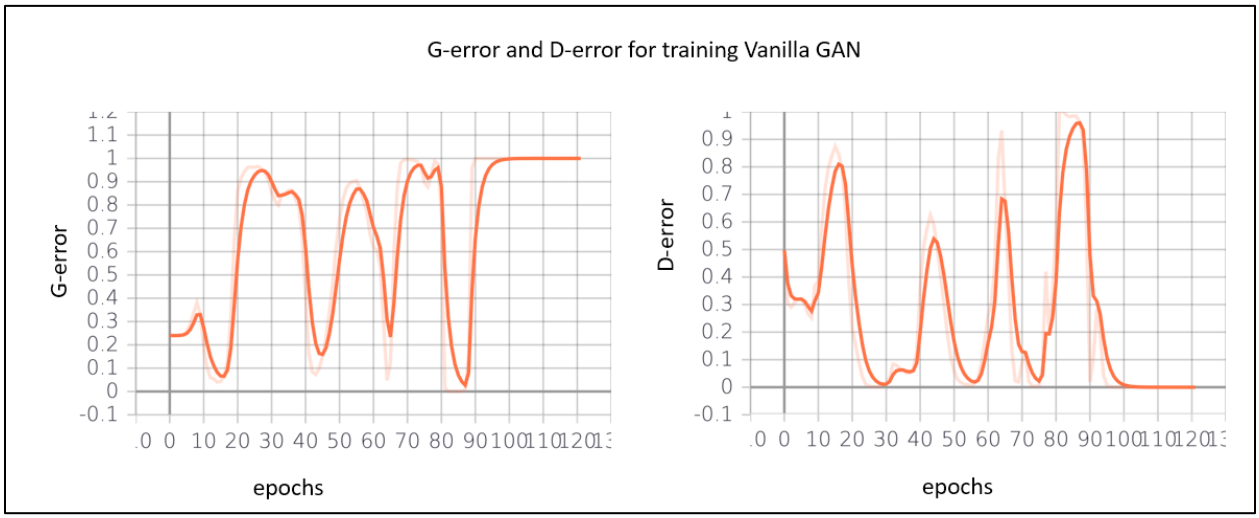


Figure 7 G-error and D-error for Training Vanilla GAN on Original Dataset

In figure 10, we show the images generated by the generator on being inputted with 16 random noise samples, as we see for each epoch the model generates the same image distribution or the same image and which is representative of mode collapse.

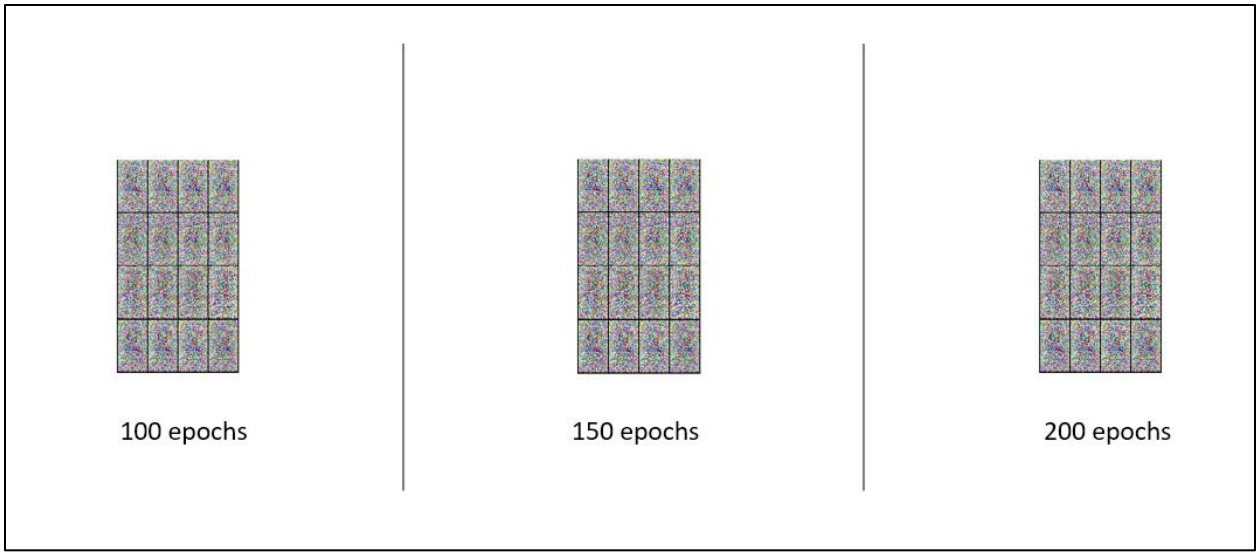


Figure 8 Test images generated for spicules at various epochs by Vanilla GAN

In figure 11 below, we show the G-error and D-error for training DCGAN on the original dataset, we notice that the error values spike quite a bit initially which is as expected. We train the DCGAN for 200 epochs or steps only due to hardware limitations. The initial results for the DCGAN seem quite promising and the adversarial network does not seem to mode collapse in the first 200 epochs.

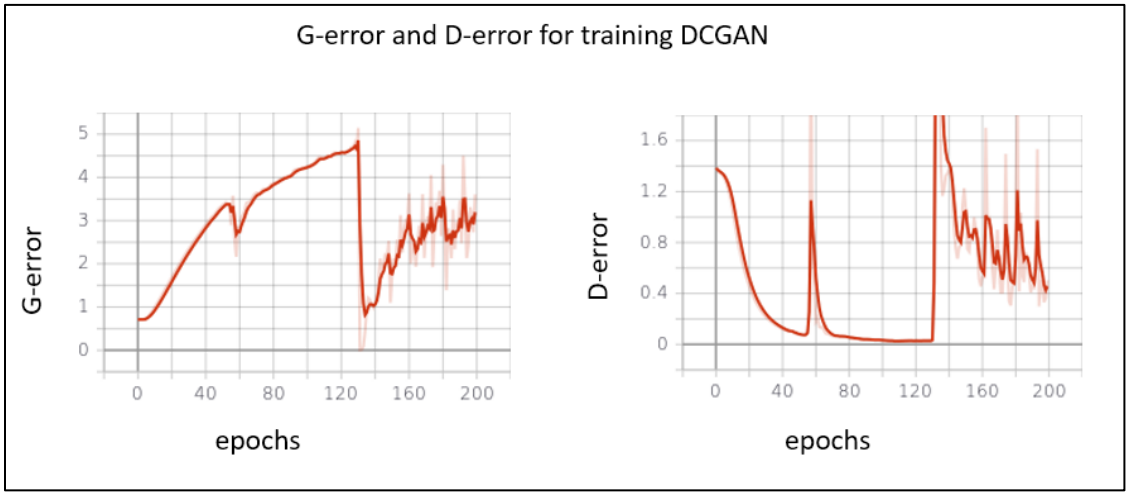


Figure 9 G-error and D-error for training DCGAN

Figure 12 below shows the test images generated by the DCGAN at various epochs, and for the 200th epoch of training the images generated visually seem to fall into a pattern of spicules.

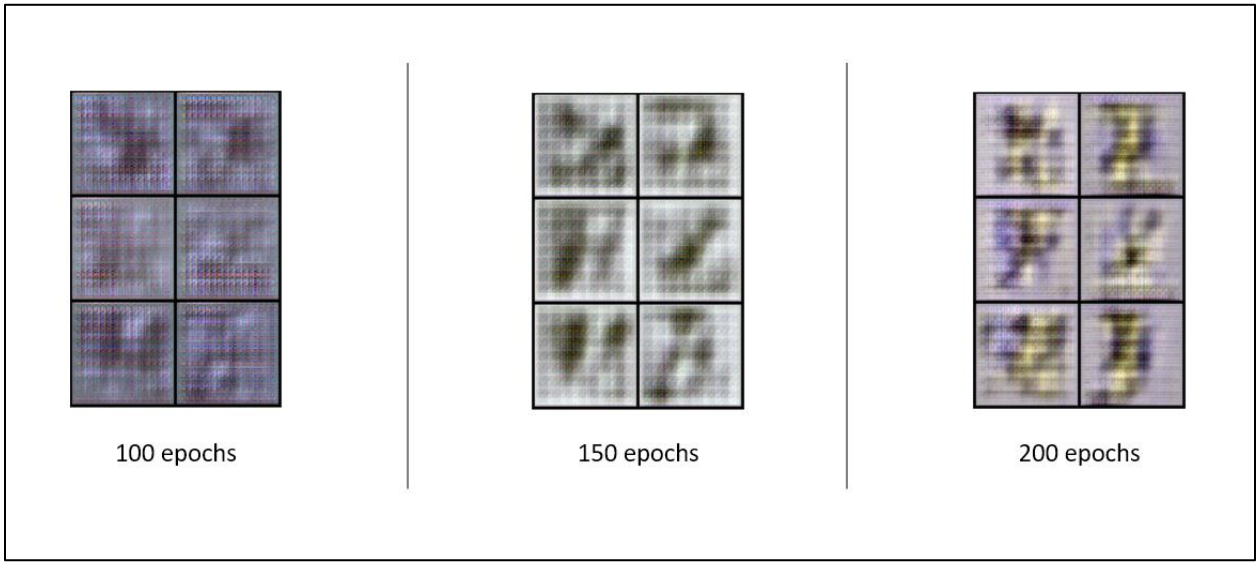


Figure 10 Images generated by the DCGAN generator on 6 randomly sampled noise inputs at various epochs

## 4.3. Performance of Classification Models

### 4.3.1. Support Vector Machine

#### 4.3.1.1. Principal Component Analysis

We perform principal component analysis as an approach to reduce the training time of the SVM model, reduce the dimensionality of the training data, avoid overfitting and increase the generalizability of the model. For choosing an appropriate dimension for the image set we calculate the explained variances of the datasets over a range of the number of components. Figure 13, below shows the plot for the explained variances vs number of components calculated for the original dataset over a range of 0 to 175 components and figure 14, shows the same for the transformed dataset.

Since figures 13 and 14, do not show a clear single point of inflection representative of the optimal number of components to choose for either of the datasets we select a range from the graphs and experimentally choose 30 as the number of components to decompose to for the original dataset and 40 for the transformed dataset as these values produce the highest accuracy, for our results.

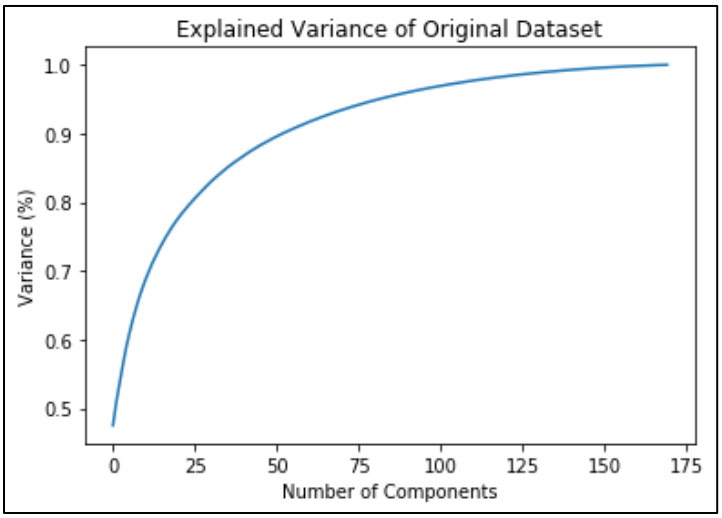


Figure 11 Explained Variance Obtained by performing PCA on the Original Training Dataset

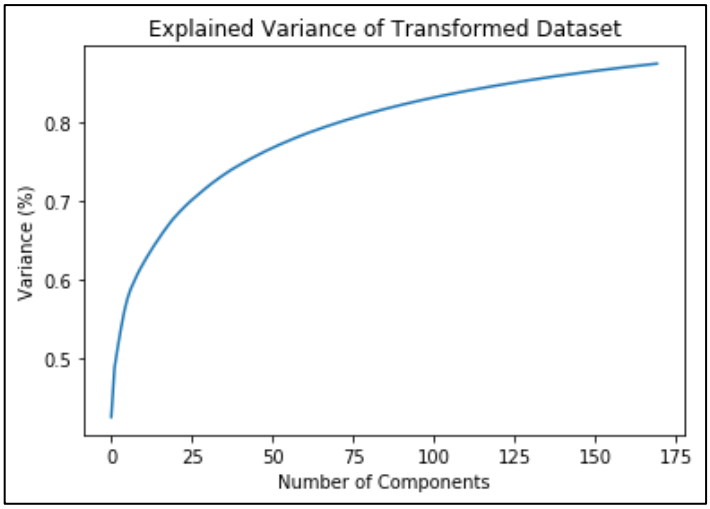


Figure 12 Explained Variance of Transformed Dataset for 170 components

We further validate our choice of the number of components for the transformed dataset by showing the explained variance vs number of components tested over a range of 0 to 500 components in figure 15 below. Figure 15 clearly estimates the point of inflection to be around 40-45 for the transformed dataset and our experimental results return the maximum accuracy for the value 40.

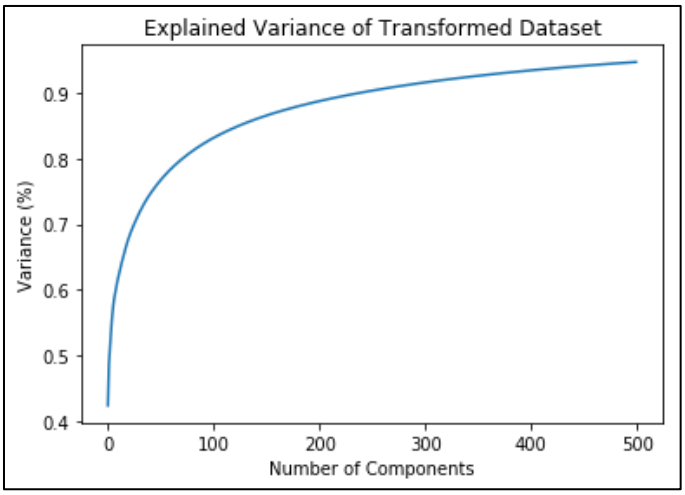


Figure 13 Explained variance obtained on the Transformed Dataset for 500 components

4.3.1.2. SVM Parameter Optimization

We use the grid search cross validation method for optimizing the C value and gamma value for the SVMs trained on the original dataset and transformed dataset. Both the C and gamma values tested are in the range $[10^{-6},10^{0}]$. Figure 16 below shows the heat-map for the validation accuracy for each combination of C and gamma value for the SVM trained on the original training set (SVM-Original) and Figure 17 shows the same for the SVM trained on the transformed training set (SVM-Transformed) the C value selected for SVM-original was 0.1 with a gamma value of 0.001 and for SVM-Transformed the C value selected was 1 and gamma value was 0.001.

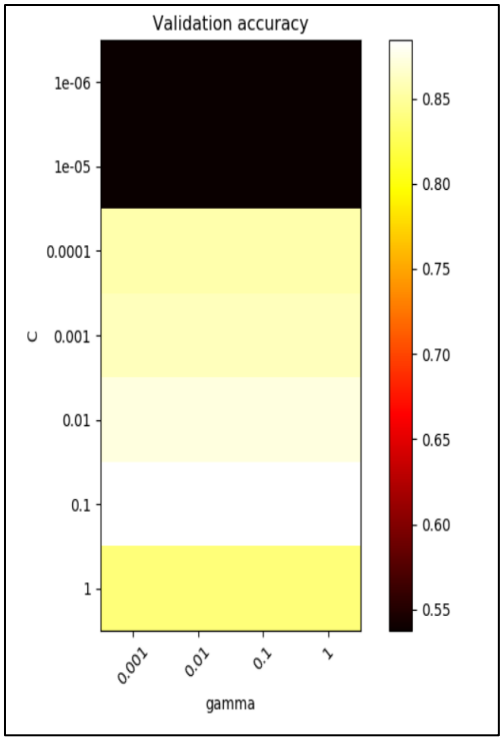


Figure 14 Heatmap for validation accuracy for C and Gamma for SVM-Original

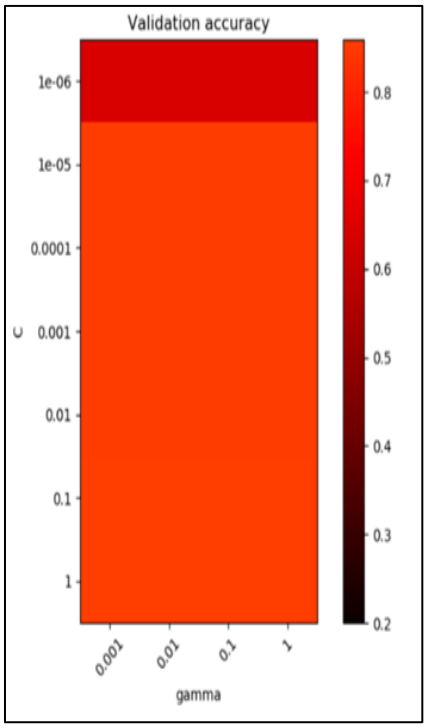


Figure 15 Heatmap for validation accuracy for C and Gamma for SVM-Transformed

### 4.3.2. CNN Training Performance

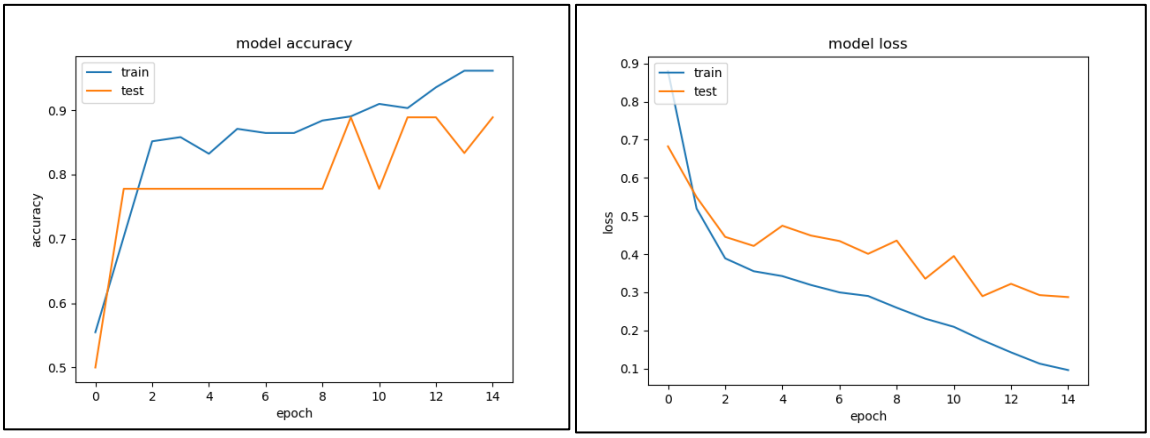


Figure 16 CNN-Original training accuracy(left) and loss(right)

We report training results for two CNN models, one trained on the original dataset (CNN-Original) and the other trained on the transformed dataset (CNN-Transformed) trained for 15 epochs. Figure 18 above, shows the training accuracy on the left and the model loss on the right along with the validation accuracy for 15 epochs for CNN-Original. The model was able to achieve a training accuracy of 96% accuracy and validation accuracy of about 90% by the end of training. Figure 19 shows the same statistics for CNN-Transformed for 15 epochs. CNN-Transformed reached a training accuracy of about 98% and a validation accuracy of about 95%.

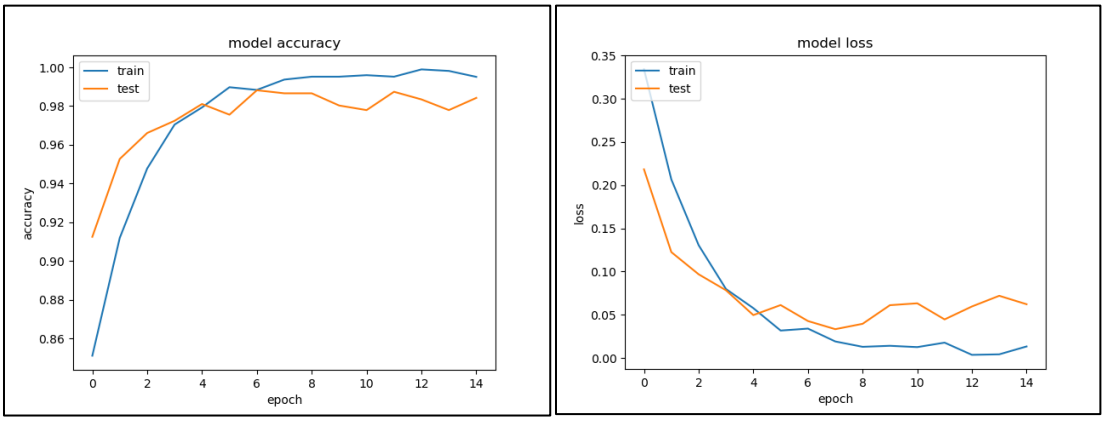


Figure 17 CNN-Transformed training accuracy(left) and loss(right)

### 4.3.3. Classification Test Results

In Table III. below we tabulate the summary of all our test results and display the test accuracy, F-score, precision and recall on the trained classification models—SVM-Original, SVM-Transformed, CNN-Original and CNN-Transformed, using the original balanced test set which was held out separately at the beginning. Using the transformed dataset to train the CNN model we achieved an accuracy of about 95%. We note that the accuracy of the SVM-Original is about 83% which is slightly higher than SVM-Transformed which returned an accuracy of 80%. Finally, the accuracy of the CNN-Original was about 91% and thus, CNN-Transformed performed much better than any of the other models.

TABLE III. CLASSIFICATION TEST RESULTS

| **Model** | **Test Accuracy** | **F Score** | **Precision** | **Recall** |
|---|---|---|---|---|
| **SVM-Original** | 0.8333 | 0.8331 | 0.8348 | 0.8333 |
| **SVM-Transformed** | 0.8000 | 0.7916 | 0.8571 | 0.8000 |
| **CNN-Original** | 0.9166 | 0.9161 | 0.9286 | 0.9167 |
| **CNN-Transformed** | 0.9500 | 0.9499 | 0.9505 | 0.9500 |

## 5. Conclusion

We have developed four different classification models and adopted a completely data driven approach for comparing the models' performances. We have also

experimented with various image transformation techniques to manipulate objects within the images and further enhance the dataset, which we experimentally validate to have a similar histogram distribution as the original dataset. In fact, figure 8 which compares the mean histogram distributions of the original and transformed image sets shows that the addition of the new images generated to the dataset smoothens out the histogram distribution and reduces spikes such that the overall distribution is normalized. We conclude that this step was imperative in increasing the performance of the CNN classifier. Performing PCA analysis on the training dataset in order to reduce training dimensionality of the SVMs reduced the training time for our SVM classifiers. The reduction in dimensionality also increased the performance of SVM-Original whereas it did not affect the performance of SVM-Transformed. We note that both the CNN models trained on the original dataset and the transformed dataset perform significantly better and reach a high-test accuracy of 95% from our benchmark SVMs. We have presented some initial results for using GANs as an approach to synthesize spicule images and by extension images of non-spicules. Out of the two types of GANs we have tested, we conclude that the DCGAN performed significantly better than the Vanilla GAN and has the potential to generate highly accurate images of spicules with more training epochs, a larger dataset and suitable computational resources. We run into the mode collapse issue in Vanilla GANs which is partly due to the small size of the dataset. In conclusion, the methods described in this report solve two major problems—firstly, we mitigate the challenges due to the small size of the dataset by adopting a highly specific approach of isolating and applying geometric transformations to the objects within the images to produce different orientations of the images and enhance the dataset from roughly 200 images to about 12000 and secondly, we develop a highly accurate CNN classifier and validate its performance against SVM which gives us about 95% test accuracy using the transformed data.

# 6. Discussion and Future Work

In our exploratory research towards increasing the size of image datasets by generating different geometric orientations of the objects within the image we find that these transformations not only enhance the dataset and distribution but also largely improve the performance of the classifiers, it would be interesting to further explore this method in two ways – (1) performing more detailed statistical analysis on the newly generated images and (2) how specific learning parameters of different classification models are affected by training on the transformed dataset. Another interesting avenue to explore for image generation would be to use the 3-D representation of images to generate images of objects with different spatial orientations. We show an initial 3-D model of a spicule in figure 24 of the appendix. For our experiments with GAN models it would be beneficial to explore allowing more training epochs and using our transformed dataset specifically for the DCGAN as an approach for artificial image synthesis. We would like to highlight that as mentioned in the previous sections the SVM-Transformed model does not realize significant improvement in performance as expected and we presume that this discrepancy could be due to choosing a sub-optimal dimension size to reduce to for the PCA analysis before training, this is another area that can be improved in terms of image pre-processing specifically for the spicule vs non-spicule dataset that we have. In terms of the overarching classification problem itself, the next step would be to develop a multiclass CNN classifier to classify spicules and other particles based on their species, this would require a large sized labelled dataset. We believe that our work can be extended greatly to further simplify the task of studying the variation in the taxa and population size of oceanic sponges through time.

This work constitutes a valuable step toward real-time shipboard analysis of sponge spicules.

# References


[1] R. C. Brusca and G. J. Brusca, Invertebrates. 3rd edition, Sunderland, MA: Sinauer Associates, 2003.

[2] P. Myers, "Porifera (On-line)," Animal Diversity Web., 2001. [Online]. Available: https://animaldiversity.org/accounts/Porifera/. [Accessed 27 04 2019].

[3] D. Lavrov, "Porifera. Sponges. Version 30 March 2009 (under construction)," he Tree of Life Web Project, http://tolweb.org/, 30 03 2009. [Online]. Available: http://tolweb.org/Porifera/2464/2009.03.30. [Accessed 27 04 2019].

[4] "Porifera: Class Hexactinellida (Glass Sponges).," The Columbia Electronic Encyclopedia. © 1994, 2000-2006 on Infoplease. © 2000-2017 Sandbox Networks, Inc., publishing as Infoplease., 04 December 2018. [Online]. Available: https://www.infoplease.com/encyclopedia/ecology/animals/invertebrates/porifera/class-hexactinellida-glass-sponges. [Accessed 27 04 2019].

[5] S. Kahn, G. Yahel, J. W. F. Chu, V. Tunnicliffe and S. P. Leys, "Benthic grazing and carbon sequestration by deep-water glass sponge reefs," *Limnology and Oceanography,* vol. 60, no. 1, pp. 78-88, 2015.

[6] J. B. Marliave, K. W. Conway, D. M. Gibbs, A. Lamb and C. Gibbs, "Biodiversity and rockfish recruitment in sponge gardens and bioherms of southern British Columbia," *Marine Biology,* pp. 156, 2247-2254, 2009.

[7] J. W. F. Chu, M. Maldonado, G. Yahel and S. P. Leys., "Glass sponge reefs as a silicon sink," *Marine Ecology Progress Series,* no. 441, pp. 1-14, 2011.

[8] E. Struyf, A. Smis, S. V. Damme, P. Meire and D. Conley, "The global biogeochemical silicon cycle," *Silicon,* vol. 1(4), pp. 207-213, 2009.

[9] M. D. Graham and et al, "High-resolution imaging particle analysis of freshwater cyanobacterial blooms," *Limnology and Oceanography Methods,* 2018.

[10] W. Bernard, D. E. Rumelhart and M. A. Lehr, "Neural networks: applications in industry, business and science," *Commun. ACM,* vol. 37, pp. 93-105, 1994.

[11] A. Krizhevsky, S. Ilya and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems,* 2012.

[12] L. Sovan and J. Guégan, "Artificial neural networks as a tool in ecological modelling, an introduction.," *Ecological modelling,* Vols. 120.2-3, pp. 65-73, 1999.

[13] G. Grinblat, L. C. Uzal, M. G. Larese. and P. M. Granitto., "Deep learning for plant identification using vein morphological patterns.," *Computers and Electronics in Agriculture,* vol. 127, pp. 418-424, 2016.

[14] D. Stephan and L.Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of biomedical informatics,* vol. 35, no. 5-6, pp. 352-359, 2002.

[15] C. Olivier, P. Haffner and V. N. Vapnik., "Support vector machines for histogram-based image classification.," *IEEE transactions on Neural Networks,* no. 10.5, pp. 1055-1064, 1999.

[16] I. Goodfellow and a. et, "Generative adversarial nets," *Advances in neural information processing systems,* 2014.

[17] B. Gary and A. Kaehler, Learning OpenCV: Computer vision with the OpenCV library., O'Reilly Media, Inc., 2008.

[18] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine,* vol. 35, pp. 53-65, 2018.

[19] A. Radford, L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434,* 2015.

[20] M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875,* 2017.

[21] E. L. Denton, S. Chintala, a. szlam and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in Neural Information Processing Systems 28,* C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 1486-1494.

[22] A. Osokin, A. Chessel, R. E. C. Salas and F. Vaggi, "GANs for Biological Image Synthesis," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[23] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS-W*, 2017.

[24] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North and G. Woodhull, "Graphviz and dynagraph – static and dynamic graph drawing tools," in *GRAPH DRAWING SOFTWARE*, Springer-Verlag, 2003, pp. 127-148.

[25] B. a. W. N. a. C. T. a. L. M. Xu, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853,* 2015.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research,* vol. 15, pp. 1929-1958, 2014.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.

[28] J. T. Springenberg, A. Dosovitskiy, T. Brox and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806,* 2014.

[29] A. Mordvintsev, C. Olah and M. Tyka, " Inceptionism: Going deeper into neural networks," Google, 17 June 2015. [Online]. Available: https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html. [Accessed 04 April 2019].

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167,* 2015.

[31] C. a. V. V. Cortes, "Support-vector networks," *Machine Learning,* vol. 20, no. 3, pp. 273-297, 01 Sep 1995.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in {P}ython," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[33] R. Sahak, W. Mansor, Y. K. Lee, A. I. M. Yassin and A. Zabidi, "Performance of combined support vector machine and principal component analysis in recognizing infant cry with asphyxia," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, 2010.

[34] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[35] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285-1298, 2016.

[36] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang and J. Cai, "Recent advances in convolutional neural networks," *Pattern Recognition,* vol. 77, pp. 354-377, 2018.

[37] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard and others, "Tensorflow: A system for large-scale machine learning," in *Symposium on Operating Systems Design and Implementation*, 2016.

# Appendix

We show 5 images (Figure 20- Figure 24) of the architecture of the VGAN, DCGAN, ConvNet and 3-D view of a spicule image.

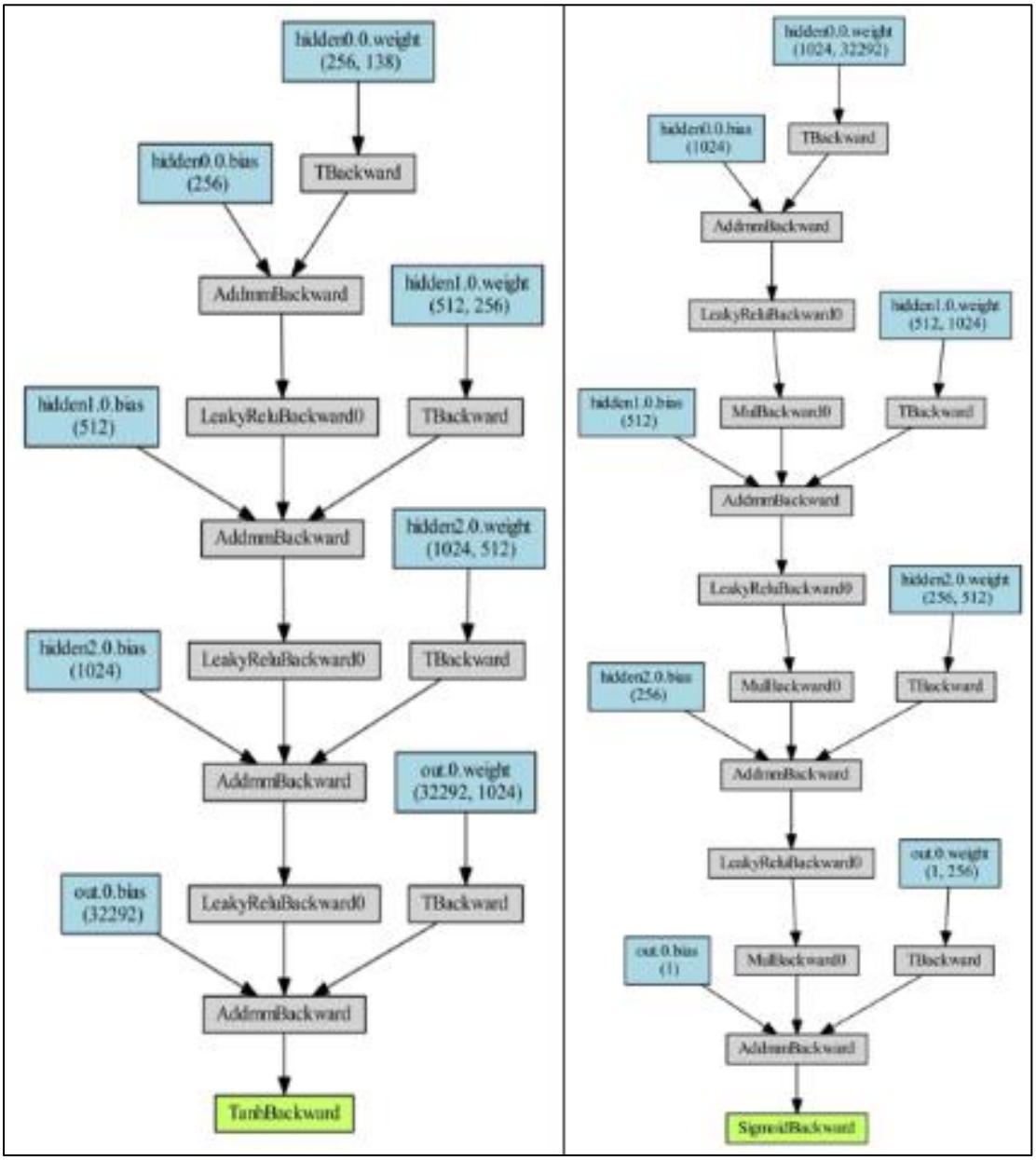


Figure 18 The architecture of Vanilla GAN - (left) Generator Network and (right) Discriminator Network

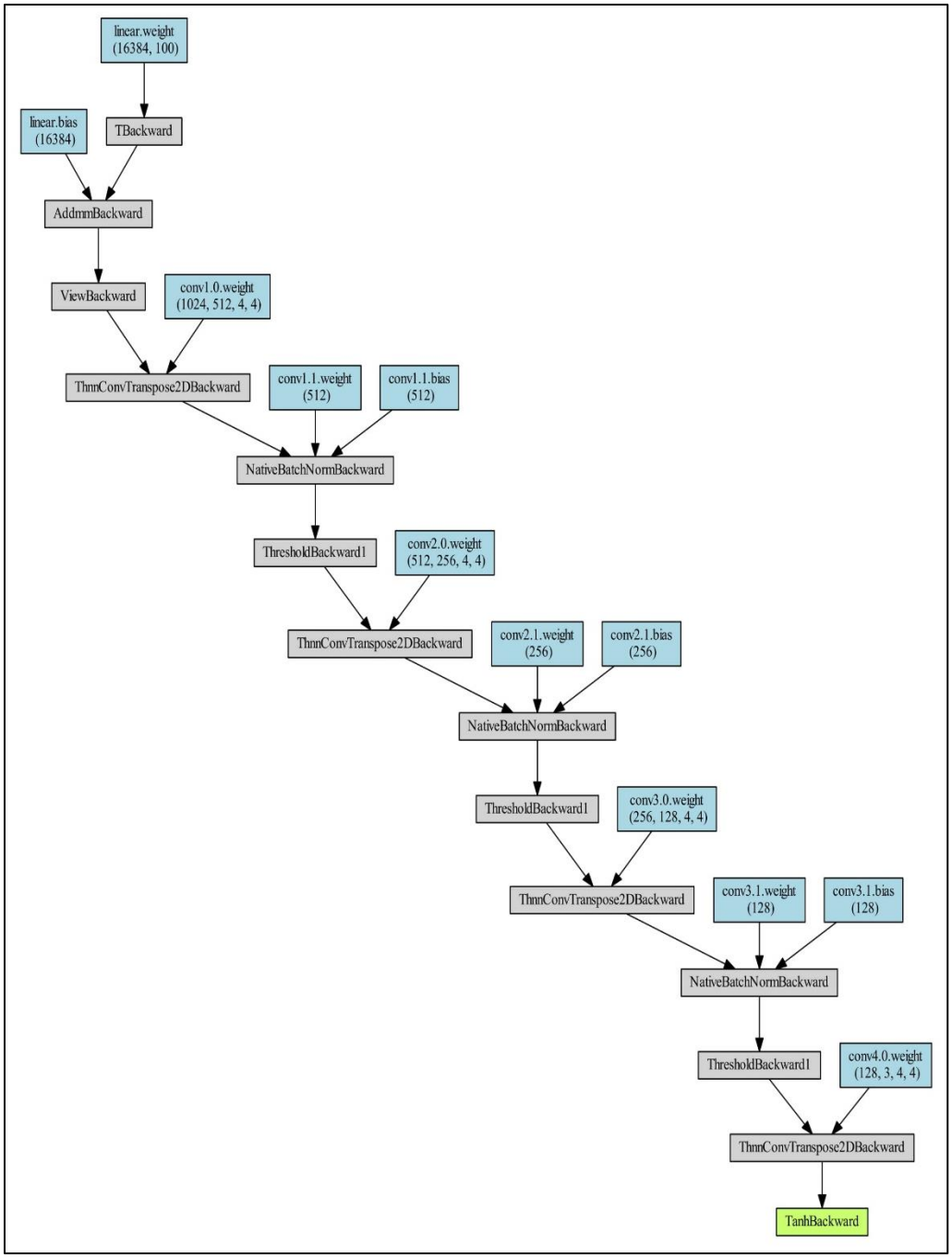


Figure 19 DCGAN Generator Network Architecture

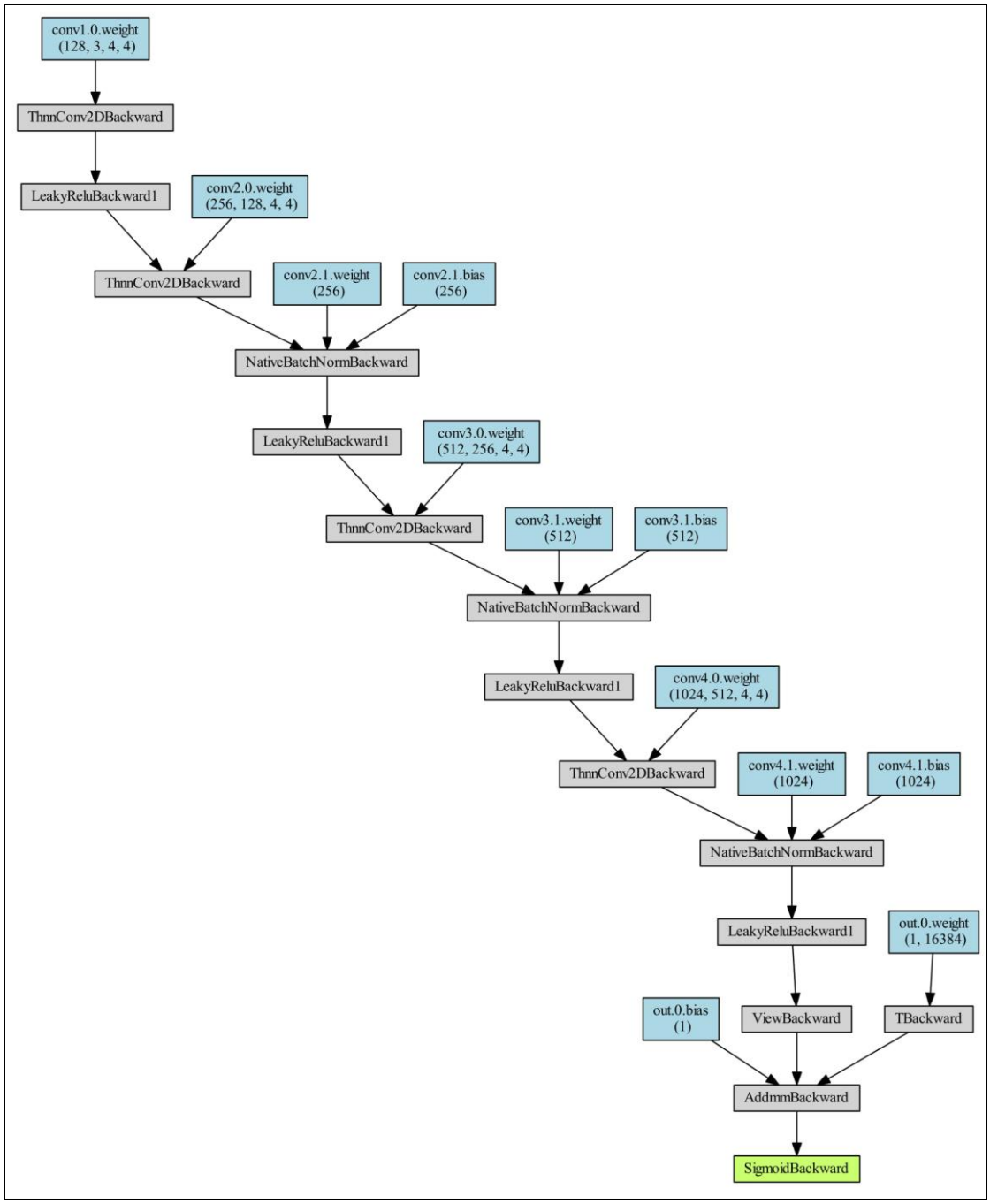


Figure 20 DCGAN- Architecture of the Discriminator

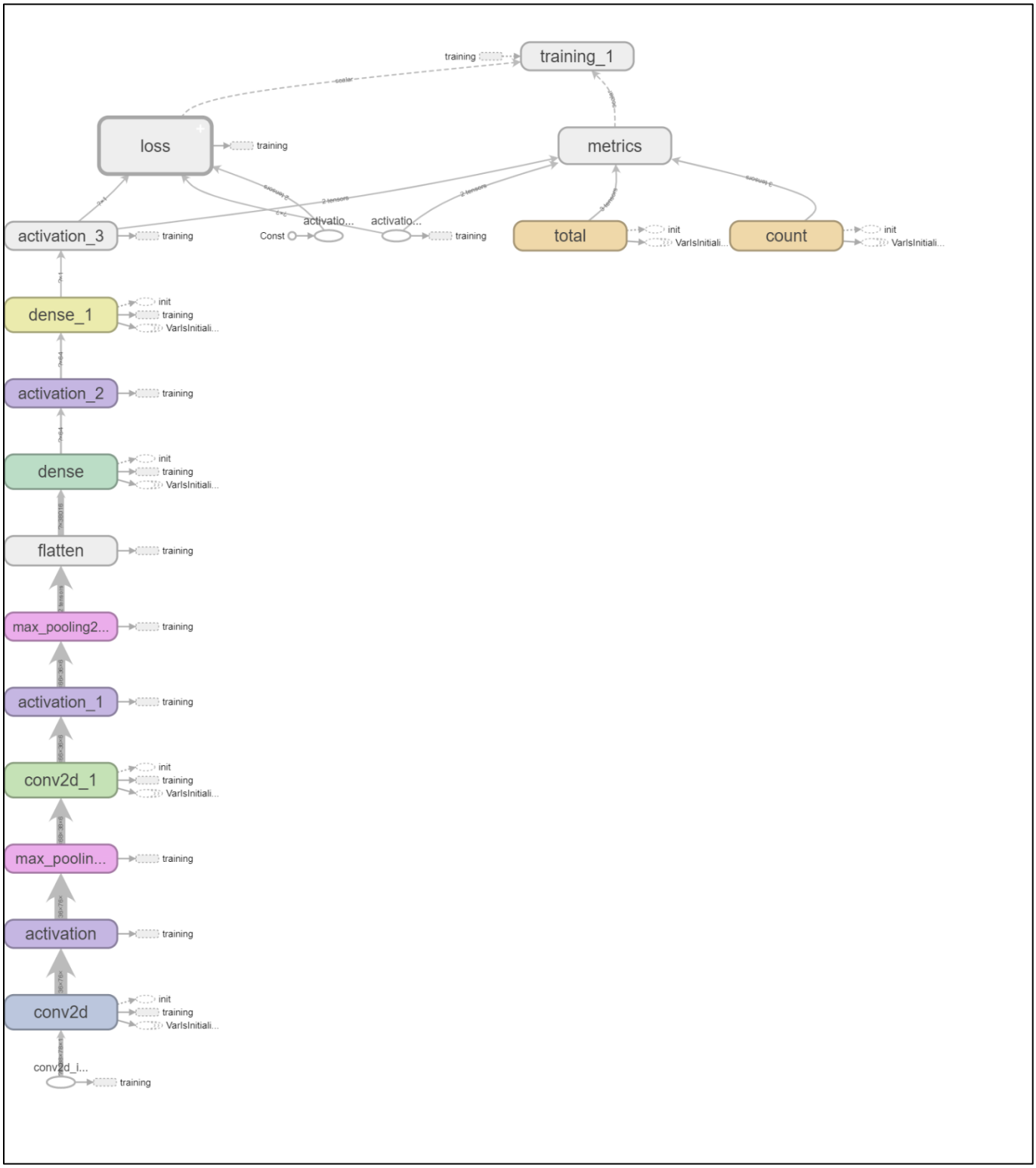


Figure 21 Architecture of the Deep ConvNet used for classification

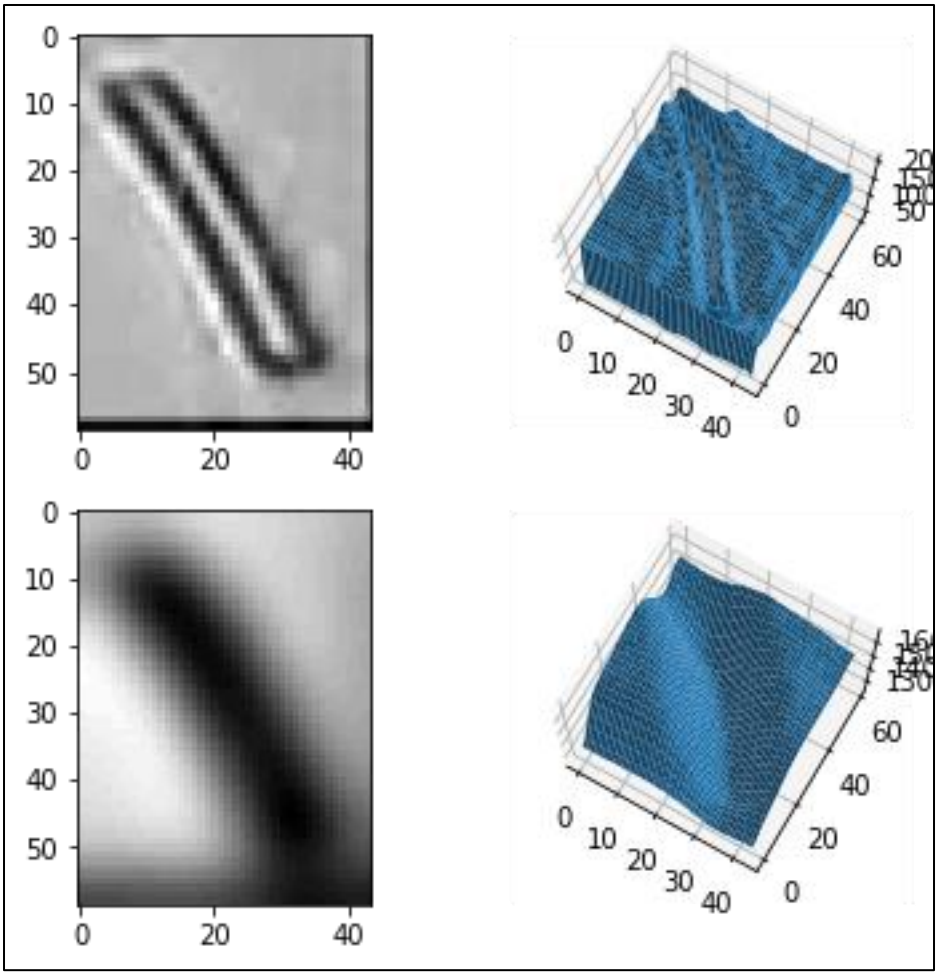


Figure 22 3-D View Generation for a Sample Spicule Image