Robust Deep Unsupervised Learning Framework to Discover Unseen Plankton Species

Eivind Salvesen[1], Aya Saad[1], Annette Stahl[1]

[1]Dept. of Engineering Cybernetics, Norwegian University of Science and Technology, NTNU, Trondheim, Norway

ABSTRACT

Deep convolutional neural networks have proven effective in computer vision, especially in the task of image classification. Nevertheless, the success is limited to supervised learning approaches, requiring extensive amounts of labeled training data that impose time-consuming manual efforts. Unsupervised deep learning methods were introduced to overcome this challenge. The gap, however, towards achieving comparable classification accuracy to supervised learning is still significant. This paper presents a deep learning framework for images of planktonic organisms with no ground truth or manually labeled data. This work combines feature extraction methods using state-of-the-art unsupervised training schemes with clustering algorithms to minimize the labeling effort while improving the classification process based on essential features learned by the deep learning model. The models utilized in the framework are tested over existing planktonic data sets. Empirical results show that unsupervised approaches that cluster the data based on the deep learning model's feature space representations improve the classification task and can identify classes that have not been seen during the learning process.

Keywords: image analysis, deep learning, plankton taxa distribution, unsupervised learning

1. INTRODUCTION

In recent years several underwater camera systems have been proposed for the image-based sampling of plankton and other microscopic particles [1–3]. These methods have made sampling easier, less time-consuming, and much more affordable than traditional sampling methods. Furthermore, continuous real-time surveillance makes it possible to develop rapid detection systems or gather long time-series of plankton data vital for advancing observations of oceanographic phenomena. However, classifying and assessing the increasing quantities of planktonic images is rapidly becoming impractical since traditional classification requires extensive labeling effort from domain experts.

Similar problems have been solved in other domains by training a machine learning (ML) algorithm to perform the image classification. Therefore, it is of little surprise that the increase in readily available planktonic image datasets has resulted in several attempts to adopt ML for computer vision in the plankton domain. In the beginning, the methods relied on careful selection of hand-designed features, which were subsequently used to train the ML classifier [4]. Even though several methods show promising results [5, 6], the challenge related to labor-intensive work for labeling partially remains. The careful selection and manual design of feature descriptors that are required to get a well-functioning ML model can be both difficult and labor-intensive [7].

In the last decade, attention has turned to deep learning models that are capable of learning good feature descriptors on their own. Convolutional Neural Networks (CNN) especially proved to be capable of detecting and classifying a wide set of plankton classes [8–11]. The recent successes are, however, tied to the supervised domain where labeled data is a necessity. Labeling effort is thus quickly getting unmanageable since deep learning models require massive training sets containing thousands of images. Machine learning can provide a robust and efficient classification of plankton at a low cost. Still, there are some drawbacks and challenges that are important to understand. Single phytoplankton objects are usually too small for image sampling methods. However, they often form algal groups such as filaments, chains, or colonies, which are easier to spot at the cost of varying size and shape characteristics.

The contribution in this paper is an unsupervised deep learning framework that combines two main functional tasks exerted sequentially. The first task is a feature extraction module based on unsupervised deep learning algorithms that produces feature space representations of the input images to the model. The second task is a clustering module that combines the feature space representations output from the first task and categorizes them into a set of classes. As part of this contribution, experimental evaluations are conducted on different unsupervised algorithms utilized in the feature extraction task and the clustering algorithms used as the last step of the framework. The main goal of the conducted evaluations is to investigate a better combination of approaches performing both tasks in an unsupervised manner.

Results show that using a rotational invariant unsupervised autoencoder is superior in extracting the feature space representations yielding a better clustering that is capable of separating objects in the given images that belong to classes never seen or learned by the model. On the other hand, unsupervised methods based on deep clustering mimic the training process utilized by the supervised learning algorithms. This training process is based on a labeled dataset produced in an unsupervised fashion. Deep clustering methods extract linear feature representations that closely match features produced by supervised deep learning methods and significantly improve the linear classifiers' performance.

The framework presented in this paper proved that using the combination of unsupervised learning models to extract the feature representation space along with a clustering algorithm can significantly improve the performance of the classification task while minimizing the exhaustive manual labeling effort required from the domain experts. Furthermore, the model can be used in real-world applications that necessitate identifying objects belonging to classes that have never been seen nor learned by the model.

2. BACKGROUND

The task of **unsupervised learning** typically observes structures or informative patterns in the data for which no prior knowledge is required. *Clustering* is one of the main unsupervised tasks aiming to find commonalities within groups of data [12]. In another use case, so-called *feature extraction*, high dimensional data is mapped down into a few dimensions [13], making it possible to capture the essence of the data with much fewer parameters [14].

Early on, the conventional approach to image feature extraction was based upon the selection of hand-crafted image attributes such as object size, shape, and color. Albeit providing adequate features for the specified task, such an approach requires carefully designed filters and is dependant on domain knowledge [15]. Furthermore, since the extraction method is specialized to a specific domain, it cannot be adapted to other tasks. Therefore, a requirement for more effective implementation and utilization of feature extraction algorithms is their ability to adapt to a wide variety of image domains [14]. The *scale-invariant feature transform* (SIFT) [16] and the *Speeded Up Robust Features* (SURF) [17] models are among the more versatile models. The SIFT and the SURF are methods that detect key points from various image locations then extract distinctive features based on the gradient around these points.

In recent years, deep artificial neural networks (ANN) have achieved excellent results on many difficult image classification tasks. Deep neural networks typically consist of several stacked layers extracting relevant information from the input before a final layer outputs a prediction. One of the biggest strengths of ANN is the capability of the first layers to learn the important structures and patterns in the input data and gradually evolve these structures into more refined concepts [7]. This learning process is called **representation learning** and similarly to a conventional feature extractor, the learned concepts can be extracted and used as features in an independent classification algorithm. For instance, in supervised deep learning models, the feature extraction is exerted by the first layers, whereas the last fully connected layer can be seen as a linear classifier. However, deep learning feature extraction as a single step is typically associated with the semi-supervised or unsupervised domain where a deep learning model is trained separately from the classifier.

A typical example of an unsupervised representational learning model is the autoencoder. An autoencoder can learn important data representations by reducing the image to a few dimensions in its *encoder* part and then reconstruct the image using a mirrored architecture of the encoder, the *decoder* part. After training, the encoder can be utilized as a feature extractor [18]. A remaining question, however, is whether such representational learning provides optimal features for classification. Hartono in [19] claimed that the "labels of the data influence the internal organization". The research suggests that models trained for classification might emphasize features to better separate groups of data. With this notion, Xie et al. [20], and Guo et al. [21] developed models which refine the autoencoder output representational space by incorporating a clustering layer into the model. Following this success, the most recent approaches to deep representational learning for classification utilize conventional neural networks trained solely on "pseudo-labels" produced by unsupervised clustering algorithms.

3. RELATED WORK

Plankton classification using unsupervised algorithms is still a relatively unexplored field of research. In 2020, Pastore et al. [22] proposed the *plankton classifier* being a set of models for unsupervised classification and detection of plankton. Their proposed data pipeline starts by reducing large images into cropped versions containing only a single planktonic organism. From these images, hand-crafted features are extracted and fed into a fuzzy-k-means clustering algorithm. This pipeline achieved well over an image data set containing 640 images, scoring an accuracy of 89%. Their approach is tested on a selected uniform set of the WHOI [10] data set, achieving a classification accuracy of 63% using a *random forest classifier*.

Comparing to the *plankton classifier* framework, a sub-goal of this paper is the extraction of features provided by a self-learned deep learning network. The unsupervised feature learning is in this regard more related to the deep learning algorithms proposed by Kyzminykh et al. [23]. Kuzminykh et al. [23] proposed a variation of an autoencoder architecture to improve image classification using their self-proposed feature extraction technique. Their framework was, among others, tested on the Kaggle data set [24]. More importantly, their objective was not to find an appropriate framework for plankton classification but to prove the efficiency of their self-proposed gram pooling technique. In this regard, the plankton data containing objects positioned with different translations and rotations proved valuable to demonstrate their framework's capability. Since their focus was not to achieve state-of-the-art results in image classification, they restricted the autoencoder to a shallow architecture. The model was first trained to learn an adequate data representation. Assuming that a better classification score is due to a better latent representation of the data, a supervised classifier was then trained on top of the feature extraction network. The overall framework showed promising results, achieving an accuracy of 62.2%, which was nearly a 10% increase to a traditional autoencoder. Furthermore, their framework highlighted the value of rotation invariant networks in regards to plankton classification.

4. PROPOSED FRAMEWORK

The proposed unsupervised classifier framework[*] categorizes input data images with no prior knowledge of the class labels. The goal is to automatically label the dataset in an unsupervised manner through a feature extraction module that conveys representative information to the clustering module, which in turn classifies the data based on identified common feature space. This approach can help discover new unseen classes since objects belonging to new classes have different feature characteristics and are likely to form new clusters. The proposed framework consists of three main components as depicted in figure 1: 1) the data preprocessing module transforms the input images to the pipeline into a consistent format, 2) the deep learning feature extraction module maps high dimensional representation into a low dimensional feature space, and 3) the classification module groups data with common feature representations into categories. The framework is designed in a highly modularized manner allowing the utilization of the different algorithm combinations suitable for the application at hand. Observe that the different components in this framework can be adopted in most object recognition algorithms. For instance, the convolutional layers of a supervised CNN network are used to extract useful features, which are then fed to the classification part represented by the last fully connected layer. However, the main purpose of this framework is the adoption of integrated approaches in the unsupervised learning domains. Feature extraction is done using a deep learning algorithm, while the classification is performed using an additional layer or a separate machine learning clustering algorithm.

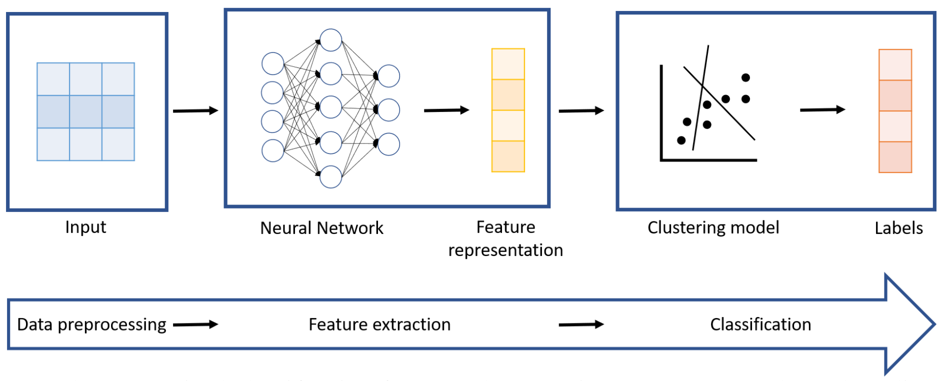

Figure 1: The proposed unsupervised classification framework. Input images are mapped to a low-dimensional space by the feature extraction network and subsequently classified by the clustering model.

[*]The code implementation can be found in the following Github repository; https://github.com/AILARON/Unsupervised-classification.

4.1 Data preprocessing module

The data preprocessing is kept simple according to the general requirement and best practices in the deep learning domain. The steps are as follows: 1. The input images are resized to 64×64 pixels. 2. The data is then centralized around zero by subtracting the mean. 3. Lastly, the data is normalized to the [-1,1] domain. During the training process, a data augmentation step is conducted to increase the model generalization and avoid overfitting. This data augmentation step applies random rotations (0-90°), horizontal and vertical of 10% shifts, zoom (0-20%), and flip (horizontal/vertical) on each batch.

4.2 Feature extraction module

For the task of image recognition, the easiest solution is to treat the input image as a vector and feed it into a machine learning algorithm. This, however, is not very effective because of the high dimensionality of real-world images. The problem of high dimensional data is referred to as *the curse of dimensionality*. First, machines typically suffer from high computational load and memory issues when trained on high-dimensional data. However, *the curse of dimensionality* is more severe because machine learning algorithms easily overfit if the number of observations is much lower than the number of input dimensions [25]. Yet, real images may contain similarities and structures such as shapes and colors, making it possible to capture the essence of the image with much fewer parameters [14]. Thus, the task of object recognition relies heavily on the machine learning algorithms' ability to extract the meaningful information from the images and discard the rest. This ability is referred to as **feature extraction**. In general, models produce either features describing the full image (global features), or features describing smaller regions or parts of the image (local features). Such choices depend on the classification, but normally global features are useful for rough segmentation and large data sets, whereas local features are used for more fine-grained classification [14].

The feature extraction module in the proposed framework is based on unsupervised deep learning algorithms to produce feature space representations of the input images. This module maps high dimensional representation into a low dimensional feature space. Two unsupervised learning approaches can be used as part of this module in alternation: the autoencoder and the deep cluster; each is described next.

4.2.1 Autoencoder model

Autoencoders have been considered one of the most promising unsupervised deep learning models. There have been several attempts utilizing them for feature extraction and classification [26, 27]. However, few successes have been achieved, and the focus has gradually shifted to purely supervised models. Autoencoders were revisited in [28], and the feature extraction capabilities of several architectures were notoriously tested. The deep convolutional models proved capable of learning data representations where similar samples were grouped and separated from dissimilar ones. However, a consistent problem was the separation of similar species of different rotations.

In this paper, the proposed architecture for the autoencoder is changed to create a more robust model for similar species with different rotations. The adopted rotation invariant autoencoder model is inspired by the work in [23] which is invariant to thirty degrees rotation. However, the backbone network utilized in [23] is relatively shallow with a different structure. Other relevant works include the supervised *Group-equivariant CNN* network introduced by Veeling et al. [29]. This network is rotation invariant to ninety degrees rotations utilizing group convolution layers. The proposed approach adopted in our framework is depicted in figure 2 where the model is trained end to end learning to reconstruct planktonic images. During the testing process, the decoder part is removed, and the latent representation learned by the encoder part is fed to a separate clustering algorithm.

The architecture is partly symmetric, using an encoder based on 5-convolutional-layer network architecture. The convolutional and transposed convolutional layers of the *encoder* are replaced by *group-convolutions* followed by group-batch normalization. The full architecture is depicted in table 1 where G-Conv layer refer to the collective group consisting of *group-convolutions*, group-batch normalization and leaky-ReLU activations. The network takes an input image of dimensions height, width and channels $H \times W \times C = 64 \times 64 \times 3$. Information is then extracted in four stages where the number of output feature maps is doubled, and the feature map size is reduced by four. Similarly, the decoder part upscales the input and reduces the dimensions using four inverse stages. The final output is a $64 \times 64 \times 3$ reconstruction transformed by a *tanh* activation function.

The architecture is constrained into an hourglass shape where input dimensions are gradually reduced to prevent model overfitting and force learning important features. The middle layer, referred to as the "bottleneck layer," consists of two layers. First, the feature maps are grouped into 512 rotation invariant feature maps using a *group-pool layer*. These maps are then transformed into a 1×512 vector by a global average pooling (GAP) layer [30]. GAP layers has proven to work well as a regularization term to prevent model overfitting and are gradually replacing traditional dense layer implementations in the recent state-of-the-art models [31]. The loss function is mean squared error (MSE), a well-attested choice to measure reconstruction loss [32]. Finally, the optimizer is chosen based on trial and error, resulting in the *Adam* [33] algorithm with standard settings.

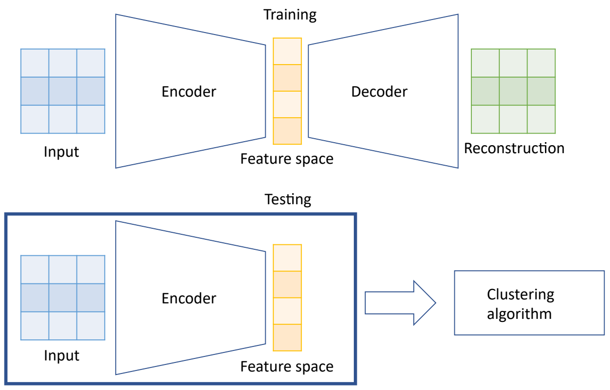

Figure 2: The autoencoder is trained by copying input to output. At test time, the decoder part is removed and the encoder is used to extract features to a separate classification algorithm.

Table 1: Overview of the rotation invariant autoencoder architecture

Layer	Kernel	Channels	Output dimension
Input	-	3	64×64
$4\times$ (G-Conv - Max pool)	(3×3 - 2×2)	256 — 512 — 1024 — 2048	64×64 — 32×32 — 16×16 — 8×8
G-Conv	3×3	2048	4×4
G-pooling	-	512	4×4
GAP	-	-	512
Dense	-	-	32768
Reshape	-	2048	4×4
$5\times$ Trans-Conv	3×3	512 — 256 — 128 — 64 — 3	8×8 — 16×16 — 64×64
Number of parameters:	45 801 795		
Loss function:	MSE		
Optimizer:	Adam		

4.2.2 Deep cluster model

One of the current state of the art models is the *Deep Cluster* [34] have achieved good performance on the ImageNet [35] data set. In the original DeepCluster work [34], Caron et al. used the AlexNet [36] and the VGG16 [37] models as backbone networks. However, the choice of the backbone network is flexible. The Deep Cluster model used in the framework is thus tested using three different CNN architectures adapted from the supervised domain: 1. 5-convolutional-layer [38], 2. VGG16 [37] and 3. ResNet [39]. The first network was chosen due to its good performance on the plankton data, while the other two have achieved excellent results on other computer vision tasks.

Training of the Deep Cluster follows the procedure listed in Algorithm 1. First, from lines 2-4, the framework is initialized. The input is a set images having input of dimensions height, width and channels $H \times W \times C = 64 \times 64 \times 3$. The model is one of the three backbone networks described above, and the clustering algorithm is k-means. Second, from lines 6-9, the pseudo-labels are created by clustering the backbone network's feature extractions. Then, from lines 11-14, the backbone network is trained on the image/pseudo-label data set for one epoch before the loop repeats. The loss function is defined as categorical cross-entropy and the optimizer is the stochastic gradient descent with a learning rate $lr = 0.005$ and momentum $m = 0.05$.

Algorithm 1 Deep Cluster

```
1: Input ← Images
2: Model ← CNN network
3: Cluster algorithm ← k-means
4: for Epoch 1 to 100 do
5:     if Get pseudo-labels then
6:         CNN ← Remove classification layer
7:         Features ← CNN(Input)
8:         Labels ← Kmeans(Features)
9:     end if
10:    if Train then
11:        CNN ← Add new classification layer
12:        dataset ← (Images, Labels)
13:        for data in dataset do
14:            Train CNN(data)
15:        end for
16:    end if
17: end for
18: return Model
```

4.3 Classification module

The last component in the proposed framework is the classification module. The input to this module is the data represented in a low-dimensional feature space. The classification module then performs clustering using one of the traditional unsupervised clustering algorithms described in this section to ensure that close features are grouped optimally and then suggests a class representing each category. The feature extraction module that adopts deep unsupervised learning approaches is a prerequisite for the classification module and significantly improves the last step of assigning the classes optimally.

k-means [40] is a partition-based clustering algorithm aiming to minimize the Euclidean distance from each sample point to one of the k cluster centers. These k clusters must be defined before the algorithm runs. The algorithm is well attested, much used, and quite fast compared to many other clustering algorithms. However, the method usually performs poorly on non-linear data, elongated clusters, or irregular shapes.

Spectral Clustering [41] (SC) is a group of more robust clustering methods, as opposed to linear methods when handling non-linear features. The algorithm can be seen as a combination of centroid- and connectivity-based clustering techniques. The algorithm first uses a connectivity-based algorithm to construct a similarity graph where each edge represents a connection between two similar points. Based on this graph, the algorithm creates a low-dimensional embedding so that connected points end up closer and non-connected points end up relatively far away. Finally, the points represented in a low dimensional space are clustered by a centroid-based clustering algorithm, the k-means.

Gaussian-mixture [13] model is a clustering algorithm that assumes the data to come from different Gaussian distributions. Clusters, typically ellipsoidal groups of points, are formed by points that are likely to have been produced by the same distribution. The algorithm starts by randomly assigning initial components (cluster centers), which are then iteratively improved using Expectation-Maximization (EM). First, the probability of each assignment being generated by each component is calculated. Then, the mean and variance of each component are tweaked to accommodate these assignments better.

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [42] is an algorithm designed to work efficiently on large data sets. The core of BIRCH clustering is to build a tree structure, called *Clustering Feature Tree*, which holds necessary information about the data, thus avoiding storing all data points. This structure can then be used to assign new points to the desired class quickly.

5. EXPERIMENTAL RESULTS

This paper explores and chooses the appropriate building blocks and their combinations to adopt them in the proposed framework to perform an optimal unsupervised classification. Therefore, the experiments conducted aim to assess the suitability of the best feature extraction models and fit these networks with an appropriate clustering algorithm.

The experiment is conducted over the Kaggle dataset [24]. First, the deep learning feature extractors are validated over a test set of unseen samples from the Kaggle data. Then, the models' ability to adapt to new unseen plankton categories is investigated. Last, the performance results of several different classification models are presented.

The feature representation of the hundred unseen images is depicted in the t-SNE visualization in figure 3. Both models seem to have generalized well and manage to group the same class species and separate the different categories. Overall, the DC model seems to produce a slightly better representation with fewer outliers. However, the set is tiny and does not contain enough samples to represent the same species' vast variation.

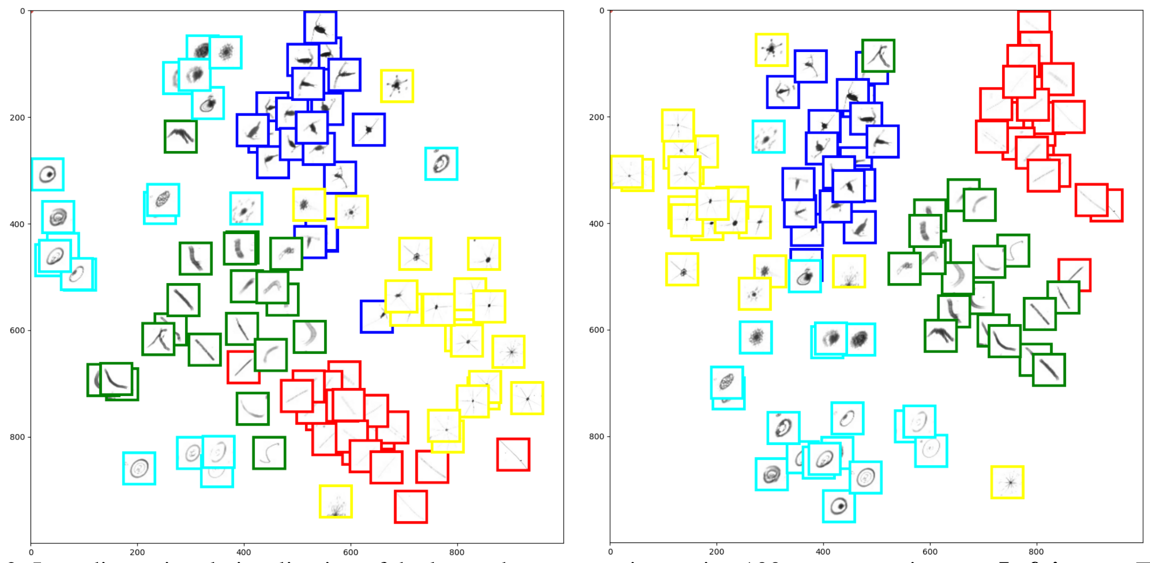

Figure 3: Low dimensional visualization of the learned representations using 100 unseen test images. **Left image:** Features extracted from the rotation invariant autoencoder model. **Right image:** Features extracted from the DC model using a VGG16 as backbone.

A similar conclusion can be drawn from the classification results given in table 2 where features are extracted by the autoencoder and DC networks and the resulting feature vectors clustered using the k-means algorithm. Observe that the clustering accuracy is higher than when clustering is performed on the seen samples from the Kaggle-DB1. This indicates that the extraction models have not overfitted to the seen data. The DC algorithm is performing slightly better because DC models produce a better linear representation. Furthermore, as seen in figure 3 the few outliers in the autoencoder representation have a significant impact when the sample size is small.

Table 2: Classification results on unseen data samples from the Kaggle data set

Model	**ACC**	**F1**	**NMI**
Auto-k-means	0.82	0.82	0.70
DC-k-means	0.93	0.93	0.86

Table 3 shows the classification results when the feature extraction models are tested over unseen data samples. The models denoted by the prefix "Missing-" refer to the deep learning feature extractor trained on a subset of the Kaggle with several classes removed. The Kaggle-5C in Table 3 is a selected subset of the Kaggle dataset consisting of five classes with different feature characteristics, and Kaggle-Full denotes the full class representations of the Kaggle dataset. All models are then tested over the whole data set, including the missing classes. Comparing the DC models, one can observe a performance decline when the model has not been trained over all classes. This indicates that the model has not learned to extract sample-specific features making the model less adaptive to new and unseen classes. The performance decline is, in comparison, very minor for the autoencoder models. This suggests that the autoencoder training method can better learn general feature representations that are easily adapted to unseen classes.

Table 3: Comparison of the feature extraction models ability to adapt to new unseen classes.

Model	**Kaggle-5C**		**Kaggle-Full**	
	ACC	**NMI**	**ACC**	**NMI**
Auto-SC	0.93	0.81	0.14	0.27
Missing-Auto-SC	0.92	0.78	0.15	0.27
DC-SC	0.90	0.80	0.10	0.26
Missing-DC-SC	0.67	0.64	0.10	0.25

Table 4 presents the classification performance of the classification models described in section 4.3 using features extracted from the Autoencoder (Auto) and the DC model. The best performance is achieved by the SC algorithm independent of the feature extraction network. The second best performer is the BIRCH algorithm. The Deep Embedded Clustering (DEC) [20] clustering approach utilized in [28] did not improve the classification results but instead worsened the learned data representation, resulting in poor performance.

Table 4: Classification results on Kaggle-5C using different clustering algorithms

Model	**ACC**	**F1**	**NMI**
Auto-k-means	0.75	0.72	0.61
Auto-SC	**0.93**	**0.93**	**0.81**
Auto-Birch	0.74	0.72	0.63
Auto-Gaussian	0.73	0.72	0.60
Auto-DEC	0.60	0.57	0.46
DC-k-means	0.77	0.76	0.71
DC-SC	**0.90**	**0.91**	**0.80**
DC-Birch	0.84	0.84	0.74
DC-Gaussian	0.81	0.81	0.71
DC-DEC	0.72	0.72	0.68

6. CONCLUSION

This paper assesses the building blocks that can contribute and form an unsupervised framework for plankton detection and classification. The work resulted in a proposed framework consisting of three components: preprocessing, feature extraction, and classification. The framework is highly modular, making it possible to test a wide variety of network architectures. This modular property encouraged implementing and validating different unsupervised methods to find the best combination of the framework's different components.

For feature extraction, two different unsupervised deep learning methods are implemented and tested. The rotation invariant autoencoder model is proposed as an improvement of the less robust autoencoders. In addition to the improved processing of similar species of different rotations, the model learned greatly improved feature representations resulting in a classification accuracy of 93% using SC over the Kaggle-DB1 data set. The DC method provides very similar classification accuracy when using a VGG16 as the backbone network and the SC algorithm scoring 92% over the same data set. However, the DC results are much better when comparing the methods using linear metrics, making the method more ideal for linear classification.

Several different clustering algorithms are then tested for classification. The best classification accuracy is obtained using SC regardless of the feature extraction network. Nevertheless, the algorithm is very time-consuming, making it less ideal in real-time applications when speed is crucial. Thus, the BIRCH algorithm might turn out to be a better option providing much faster predictions at the cost of slightly lower accuracy. However, the models provide robust results only when the data contains a few species and failed when predicting several more categories.

Overall, the proposed framework shows high potential and can be a valuable classification tool when the size of the data set containing ground truth labels is small. The framework shows promising application as a support classification tool that can be utilized by domain level experts and help speed up the labeling process as well as discover new classes the model has not previously encountered.

ACKNOWLEDGMENTS

This research is funded by the Research Council of Norway (RCN) IKTPLUSS program (project number 262741) and supported by NTNU AMOS (RCN project number 223254).

REFERENCES

[1] R. K. Cowen and C. M. Guigand, “In situ ichthyoplankton imaging system (isiis): system design and preliminary results,” Limnology and Oceanography: Methods **6**(2), 126–132 (2008).

[2] E. J. Davies, P. J. Brandvik, F. Leirvik, and R. Nepstad, “The use of wide-band transmittance imaging to size and classify suspended particulate matter in seawater,” Marine pollution bulletin **115**(1-2), 105–114 (2017).

[3] R. J. Olson and H. M. Sosik, “A submersible imaging-in-flow instrument to analyze nano-and microplankton: Imaging flowcytobot,” Limnology and Oceanography: Methods **5**(6), 195–203 (2007).

[4] P. González, A. Castaño, E. E. Peacock, J. Díez, J. J. Del Coz, and H. M. Sosik, “Automatic plankton quantification using deep features,” Journal of Plankton Research **41**(4), 449–463 (2019).

[5] H. M. Sosik and R. J. Olson, “Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry,” Limnology and Oceanography: Methods **5**(6), 204–216 (2007).

[6] H. Zheng, R. Wang, Z. Yu, N. Wang, Z. Gu, and B. Zheng, “Automatic plankton image classification combining multiple view features via multiple kernel learning,” BMC bioinformatics **18**(16), 570 (2017).

[7] I. Goodfellow, A. Courville, and Y. Bengio, Deep learning, Adaptive computation and machine learning, MIT Press, Cambridge, Mass (2017).

[8] X. Li and Z. Cui, “Deep residual networks for plankton classification,” in OCEANS 2016 MTS/IEEE Monterey, 1–4, IEEE (2016).

[9] J. Y. Luo, J.-O. Irisson, B. Graham, C. Guigand, A. Sarafraz, C. Mader, and R. K. Cowen, “Automated plankton image analysis using convolutional neural networks,” Limnology and Oceanography: Methods **16**(12), 814–827 (2018).

[10] E. C. Orenstein, O. Beijbom, E. E. Peacock, and H. M. Sosik, “Whoi-plankton-a large scale fine grained visual recognition benchmark dataset for plankton classification,” arXiv preprint arXiv:1510.00745 (2015).

[11] A. Saad, E. Davies, and A. Stahl, “Recent advances in visual sensing and machine learning techniques for in-situ plankton-taxa classification.” presented at Ocean Sciences Meeting 2020, San Diego, CA, 16-21 Feb. (2020). 636384.

[12] S. Russell and P. Norvig, Artificial intelligence: a modern approach, Prentice Hall series in artificial intelligence, Pearson Education (2016).

[13] C. M. Bishop, Pattern recognition and machine learning, springer (2006).

[14] A. I. Awad and M. Hassaballah, “Image feature detectors and descriptors,” Studies in Computational Intelligence. Springer International Publishing, Cham (2016).

[15] J. Yang, D. Parikh, and D. Batra, “Joint unsupervised learning of deep representations and image clusters,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5147–5156 (2016).

[16] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” International journal of computer vision **60**(2), 91–110 (2004).

[17] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in European conference on computer vision, 404–417, Springer (2006).

[18] J. Almotiri, K. Elleithy, and A. Elleithy, "Comparison of autoencoder and principal component analysis followed by neural network for e-learning using handwritten recognition," in 2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT), 1–5, IEEE (2017).

[19] P. Hartono, "Mixing autoencoder with classifier: conceptual data visualization," arXiv preprint arXiv:1912.01137 (2019).

[20] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in International conference on machine learning, 478–487 (2016).

[21] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in International conference on neural information processing, 373–382, Springer (2017).

[22] V. P. Pastore, T. G. Zimmerman, S. K. Biswas, and S. Bianco, "Annotation-free learning of plankton for classification and anomaly detection," Scientific reports **10**(1), 1–15 (2020).

[23] D. Kuzminykh, D. Polykovskiy, and A. Zhebrak, "Extracting invariant features from images using an equivariant autoencoder," in Asian Conference on Machine Learning, 438–453 (2018).

[24] R. K. Cowen, S. Sponaugle, K. L. Robinson, J. Luo, and C. Guigand, "Planktonset 1.0: Plankton imagery data collected from f.g. walton smith in straits of florida from 2014-06-03 to 2014-06-06 and used in the 2015 national data science bowl (nodc accession 0127422). national oceanographic data center, noaa. dataset.," (2015).

[25] E. R. Davies, Computer vision: principles, algorithms, applications, learning, Academic Press (2017).

[26] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in Proceedings of the fourteenth international conference on artificial intelligence and statistics, 215–223 (2011).

[27] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in 2007 IEEE conference on computer vision and pattern recognition, 1–8, IEEE (2007).

[28] E. Salvesen, A. Saad, and A. Stahl, "Robust methods of unsupervised clustering to discover new planktonic species in-situ," in OCEANS 2020/IEEE SINGAPORE, IEEE (2020).

[29] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling, "Rotation equivariant cnns for digital pathology," in International Conference on Medical image computing and computer-assisted intervention, 210–218, Springer (2018).

[30] M. Lin, Q. Chen, and S. Yan, "Network in network," arXiv preprint arXiv:1312.4400 (2013).

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

[32] A. Creswell, K. Arulkumaran, and A. A. Bharath, "On denoising autoencoders trained to minimise binary cross-entropy," arXiv preprint arXiv:1708.08487 (2017).

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

[34] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in Proceedings of the European Conference on Computer Vision (ECCV), 132–149 (2018).

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE conference on computer vision and pattern recognition, 248–255, Ieee (2009).

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Communications of the ACM **60**(6), 84–90 (2017).

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556 (2014).

[38] A. Saad, A. Stahl, A. Våge, E. Davies, T. Nordam, N. Aberle, M. Ludvigsend, G. Johnsen, J. Sousa, and K. Rajan, "Advancing ocean observation with an ai-driven mobile robotic explorer," Oceanography **33**, 42–51 (2020).

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 770–778 (2016).

[40] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, 281–297, University of California Press, Berkeley, Calif. (1967).

[41] U. Von Luxburg, "A tutorial on spectral clustering," Statistics and computing **17**(4), 395–416 (2007).

[42] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," ACM sigmod record **25**(2), 103–114 (1996).