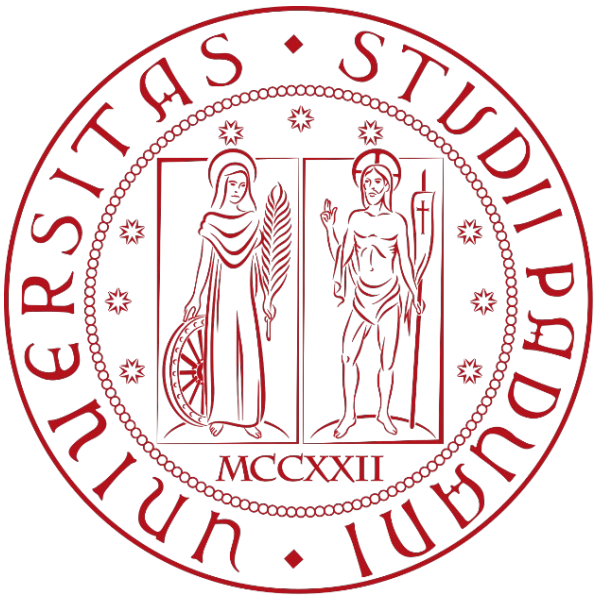
Encode information from *n* grayscale images to a single RGB image for training a CNN

Formaggio Alberto, Girotto Pietro, Orlandi Niccolò



University of Padua

Artificial Intelligence Project

Abstract

Picking foraminifera samples is as important as tedious for bio-marine research.

Given the fact that it is suitable for automation, our goal was to improve accuracy of those automated methods thanks to ways of processing images.

Multiple new strategies were implemented: Averaged percentile, Gaussian, Clustering, Weighted Clustering.

Starting from 16 different images of the same sample we used them to create a single RGB image to later train a CNN and compare our new accuracy results with the one originally implemented.

Even though our methods were not as accurate as Percentile by themselves, using ensemble strategies we were able to marginally increase the overall accuracy.

1. Introduction

1.1 Foraminifera

Foraminifera are single-celled protists secreting a tiny shell (called “test”) which is between half and one millimetre long. They can be mainly found in marine environments.

The study of fossil foraminifera has various applications and can provide information in many different fields: they provide evidence of the age of marine rocks along with evidence about past environments but they can also help us to find oil, just to mention a few usages of these micro-organisms[1].

Therefore, foraminifera retrieval is an essential but repetitive, low-reward task whose automation could really help the scholars in their research activity.

1.2 AlexNet

AlexNet is a convolutional neural network (CNN) 8 layers deep.

It was trained using more than 1 million labelled images from the ImageNet database.

The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images.

The network has an image input size of 227-by-227[2].

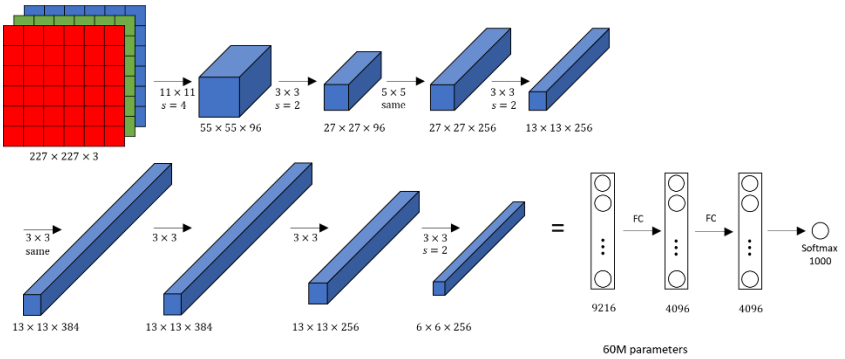


Figure 1: Representation of the workflow of AlexNet

1.3 Transfer Learning

Training a neural network can be a highly demanding task both for hardware and time requirements. Moreover, training a CNN from scratch requires huge datasets and it is not rare the case in real-world scenarios where collecting data for training a network can be difficult for several reasons. Therefore, there is the need to create high-performance learners trained on data which are easier to obtain from

different domains and then tune those learners on the task of interest. This approach of transferring knowledge is called transfer learning[4].

In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task[3].

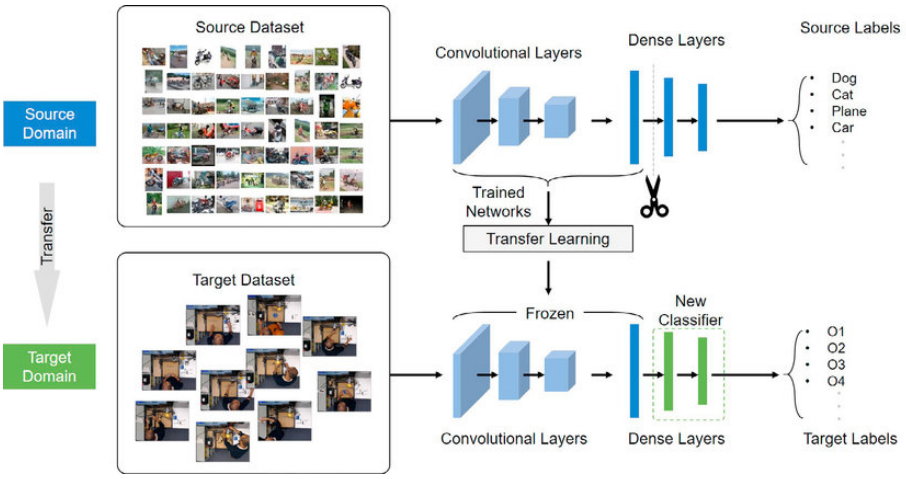


Figure 2: Example of transfer learning

2 Materials and Methods

2.1 Image Collection and identification

The collection of the images that will be used as dataset, as described in [7], were collected by using a microscope. The collected images were in shades of grey with a resolution of approximately 450x450 pixels.

A dataset of 1437 foraminifera, available at https://doi.pangaea.de/10.1594/PANGAEA.897873 , was photographed for this study.

2.2 Grayscale to RGB conversion

Due to the tools employed in the problem of interest, it was only possible to collect images in grayscale. However, light is needed in order to collect pictures and both the direction and the angle of the laser beam influence strongly the result of the taken picture by creating shadow and light zones. Therefore, clearly, collecting just a picture per sample is not enough.

What was done, then, was collecting 16 different pictures in grayscale of the organism by varying the position of the light source.

By using all the information provided by the 16-channel grayscale images of a single organism, we can create a 3-channel RGB image that we can use to train our CNN, as the network itself requires colour images as input.

2.3 Training and Testing

When training a neural network with a limited amount of pattern, there may be the risk for the network to overfit the data. Basically, the network learns very well how to classify the patterns in the training set, by almost memorizing them, but when it comes down to recognize a new pattern, the network does not scale well.

An approach useful when the number of patterns available is limited, and which we are going to use, is the k-fold cross validation.

This method is very popular due to its ability to generate a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split[5].

In k-fold cross-validation, the available learning set is divided into k subsets (the “folds”). This partitioning is performed by randomly sampling cases from the learning set without replacement.

We are going to use k-1 of these folds to train our network and the remaining one (denoted as validation set) will be used for testing purposes.

This procedure is repeated until each of the k subsets has been used as validation set.

Finally, the average of the k performance measurements on the k validation sets is the cross-validated performance that will be considered as the actual accuracy thanks to its less biased characteristics[6].

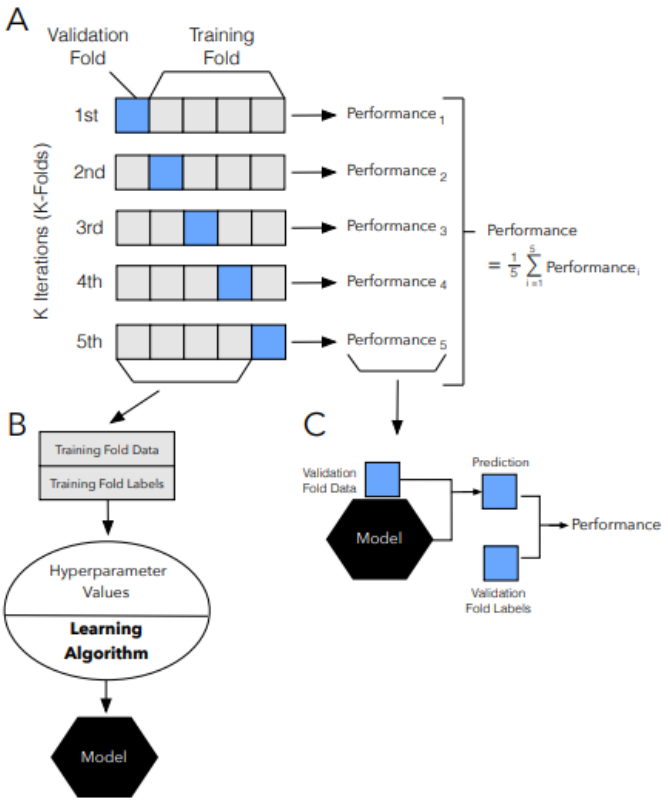


Figure 3: Illustration of the k-fold cross validation procedure

In our case, we are going to set $k = 4$. Therefore, in each iteration there will be 3 sets used for training the network and 1 as validation set.

3 Experiments

In this section, we are going to explore the several approaches that were considered in order to produce a new method to transform grayscale images into RGB ones.

3.1 Percentile

The percentile was the first approach proposed by the writers of the original paper 7].

The colour image was created by using 3 out of the 16 images applying the concept of percentile, by considering the 10^{th} percentile, median and 90^{th} percentile and mapping these images into the red, green and blue channels, respectively, to generate a single RGB image.

3.2 Averaged Percentile

The first idea we had was to modify the original percentile by extracting more features from the images in different levels.

For instance, we evaluated the average between the first 5 images and mapped those into the red channel. The following 5 were mapped into the green channel and the remaining ones into the remaining channel.

The same approach was applied also considering only 3 images instead of 5.

We decided to go down this path because in the resulting image by using the stand-alone percentile, there were several noisy pixels: pixels in the background that, instead of being black, were coloured. Using this approach, our hope was to reduce the influence of some of those dirty pixels in the resulting image i.e. if in a picture a background pixel has a value of 40 and in the remaining 4 it is considered as a 5, the average used in the final image would have been 12 instead of 40.

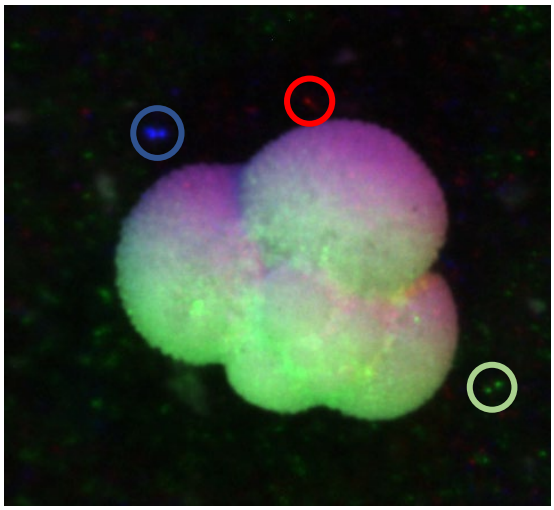


Figure 4: Example of noisy pixels in the background when using the original percentile

3.3 Gaussian

Even though the average percentile was indeed smoother, it was prone to create strong and bright images which hardly reflected the actual samples. Consequently, we turned to a Normal Distribution, in order to extrapolate the distribution of the pixels, and then used the distribution itself to colour the final image.

Specifically, for each pixel of the final image:

-all 16 samples were used to fit a normal distribution

-the green channel was the mean of the distribution

-the red channel was 2 standard deviation lower than the mean

-the blue channel was 2 standard deviation higher than the mean

The aim of this method was once again to reduce images' noisiness, overall the results were quite polished and far more believable than the averaged percentile.

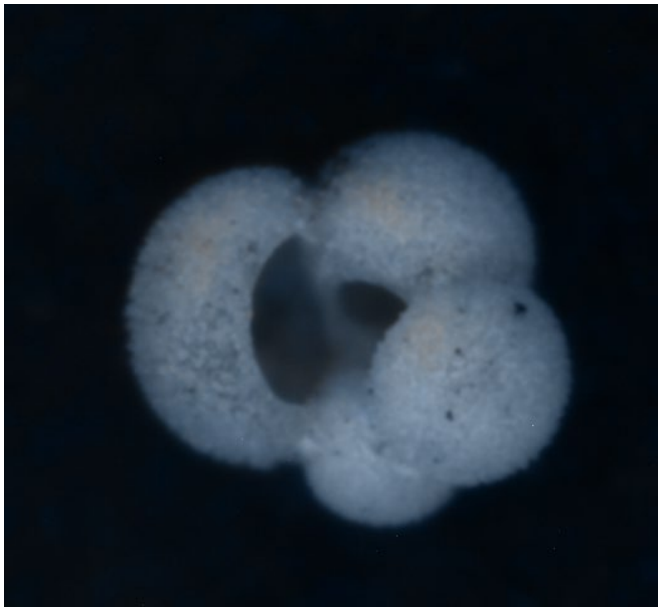


Figure 5: Sample of Gaussian processing output

3.4 Clustering

Following the gaussian processing, we thought that rather than operating directly on single pixels we could somehow group them together to extrapolate more meaningful behaviours.

For each of the pixels of the output image we used the k-means algorithm on the 16 original pixels in order to differentiate them into 3 classes. We then proceeded to calculate the average of each class and the highest mean value was assigned to the red channel, the 2nd one to the green channel and the last one to the blue channel.

Results were not as homogeneous as expected, images were extremely noisy and slightly coloured.

As a further improvement, we decided to make all the pixels for which the highest centroid value was lower than a threshold t (in this case $t = 20$) equal to 0.

We had the following idea: if, among all the clusters, the highest value for a centroid is very low, then the pixel is very dark, therefore there must have been some noise in the measurement. This pixel will be considered as black, i.e. all the channels will be set to 0.

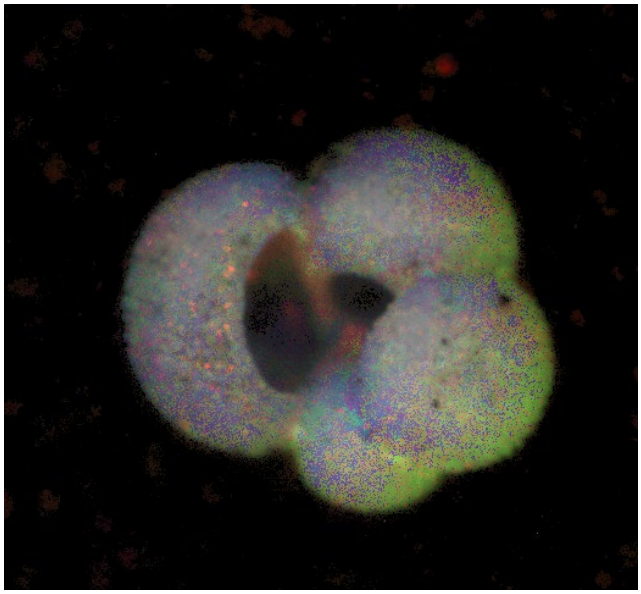


Figure 6: Sample of clustering output

3.5 Weighted Clustering

An interesting observation was done while applying the clustering: in many cases a cluster was made up of just one of the 16 pixels in the original image, therefore it was not very appropriate to weigh that cluster in the same way as a cluster containing 8 pixels.

To stem the problem, we thought it was a good idea to weigh differently clusters according to the number of patterns belonging to each cluster.

This was the solution we developed for a single pixel:

1. Count the number of patterns p_i inside the i-th cluster and sort the clusters according to p_i.
2. Consider $P = \underset{i}{max}\, p_i$ the highest number of elements in a cluster and $I = \underset{i}{argmax} p_i$ the cluster containing the maximum number of elments
3. Naming c_i the centroid of the cluster i, we copy without any change c_I in the appropriate channel in the resulting image.
4. For $i \neq I$, we evaluate the weights for the centroids c_i in the following way:

$$w_i = \frac{p_i}{P}$$

5. If the weight is lower than a given threshold T (the best performance was achieved by considering $T = 0.66$), then the weight is capped at 0.66.

Therefore:

$$w_i = max\{\frac{p_i}{P}, 0.66\}$$

6. Finally, for the given pixel, the value for the remaining two channels will be:

$$x_i = c_i * w_i$$

For deciding which centroid will be assigned to which channel we applied the same idea that was used for the percentile: the centroid with the highest value will be assigned to the blue channel, the one with the lowest value to the red one and the remaining centroid will be used for the green channel.

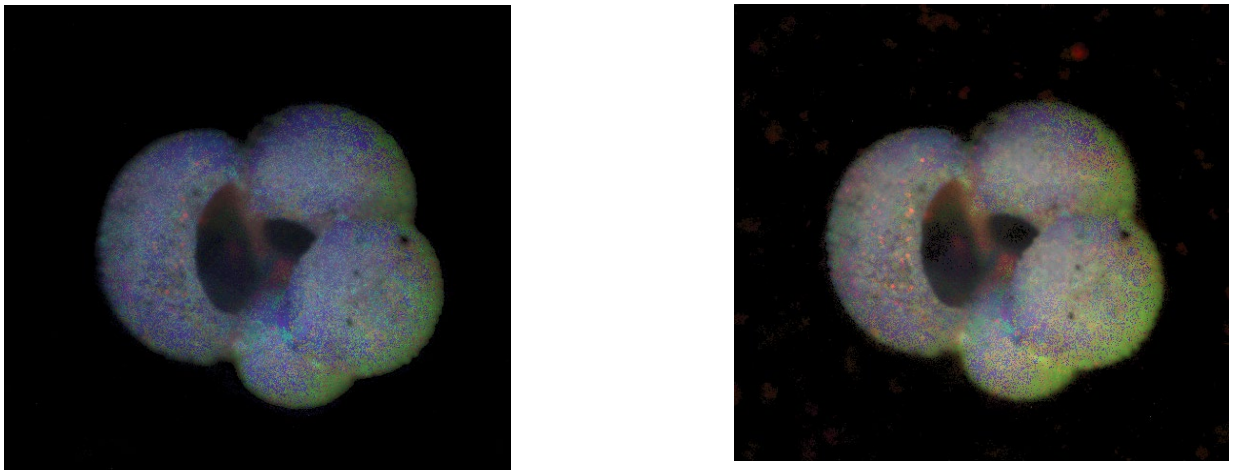


Figure 7: Resulting image with the weighted clustering approach (LEFT), comparison to simple clustering (RIGHT)

Further unsupervised dimensionality reduction methods were used in the process, hoping to get better scores: PCA, MDSI and DCT. Only the latter resulted in comparable performances, while the formers had quite low accuracy.

4 Results

In this section we are going to analyse the results of our experiments before using a standalone AlexNet for each method and later an ensemble of networks trained with the different methods available.

4.1 Standalone AlexNet

Our initial approach was to create images and do training and testing on only one AlexNet to see the performances of the classification of foraminifera by using the images produced by our methods compared to the one proposed in the original paper [7].
The accuracies of this experiment are reported in Table1, compared to the one from the literature.

Table1. Accuracy on AlexNet by using images coloured with different methods

Method	**Accuracy**
Percentile [7]	**0,8050**
Percentile Avg (1-5 6-10 11-16)	0,7938
Percentile Avg (2-4 7-9 12-14)	0,7950
DCT	0,7415
PCA	0,6933
MDSI	0,7135
Clustering	0,7306
Gaussian	0,7360
Weigthed Clustering	0,7329

The results of our new methods did not produce any improvement compared to the state-of-the-art approach.
Therefore, we decided to create an ensemble as described in the following section.

4.2 Ensemble approach

After realizing that the results were not what we expected, we decided to create an ensemble of networks by training them using images obtained by several methods.
By doing so, the diversity of our classifier improved as the features were extracted in different ways with the hope to improve the overall accuracy.
The ensembles that we developed are the following:

- Ensemble 1: The first ensemble was composed by five networks respectively based on Percentile, DCT, clustering , Gaussian and weighted clustering methods.
- Ensemble 2: The second ensemble was formed by five networks based on the state-of-the-art approach: we therefore used 2xPercentile, 2xPercentile(2-4) and 1xPercentile(1-5). The classifiers in this ensemble are highly correlated, but we wanted to see if any improvement was possible by using different methods relying on the same winning strategy.
- Ensemble 3: We now decided to use a less correlated ensemble by still weighing more the Percentile by using the 3 methods we developed and the two best performing methods developed by us.

  Therefore, we have: Percentile, Percentile(1-5), Percentile (2-4), Weighted Clustering, Gaussian
- Ensemble 4: The same as Ensemble 3, but with normal clustering instead of weighted clustering. Percentile, Percentile(1-5), Percentile (2-4), Clustering, Gaussian
- Ensemble 5: Percentile, Percentile (2-4), Clustering, Gaussian, DCT
- Ensemble 6: Percentile, Percentile (2-4), Weighted Clustering, Gaussian, DCT

Where Percentile is the standard approach, Percentile(1-5) uses 5 pixels to determine each RGB colour and Percentile(2-4) uses 3 pixels. This way, we have the same images with different brightness. The reason why some networks were used more times is to weigh more the best performing methods. As it is noticeable form the ensembled afore reported, we tried also to compare the performance of Clustering against its weighted version.

The results of this experiment are reported in Table2.

Table 2. Results of ensemble approach

Ensemble	**Accuracy**
Ensemble 1	0,8269
Ensemble 2	0,8190
Ensemble 3	0.8331
Ensemble 4	**0.8525**
Ensemble 5	0.8520
Ensemble 6	0.8323

We can see that Ensemble 4 is the best performing ensemble by improving of about 5% the performance of the standalone percentile method.

Anyways, even the worst performing ensemble still had an improvement of 1% over the state-of-the-art method in the standalone version.

Another interesting observation can be done by analysing the performance of ensembles 3-4 and 5-6: without changing the other 4 methods, the simple clustering method has outweighed the performance of its weighted version even though the latter had a slightly better performance when performing on its own. This could be due to the fact that clustering brings different information compared to weighted clustering.

5 Conclusions

The main goal has been reached, the accuracy has been improved using a different method than those presented in the initial paper.

Unfortunately, the methods that were developed by us were not as good as the original percentile but we have still been able to improve the overall performance by using an ensemble.

As a future work, an optimal number of classifiers can be found, the Q-statistic can be evaluated to determine the correlation between the classifiers and find the most uncorrelated methods to use in the ensemble.

Moreover, the k-fold cross validation can be extended (we used 4-fold cross validation but the training set could have been split in even more folds).

6 References

[1] https://ucmp.berkeley.edu/fosrec/Wetmore.html

[2] https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[3] https://arxiv.org/abs/1411.1792

[4] https://journalofbigdata.springeropen.com/articles/10.1186/s40537-016-0043-6

[5] https://machinelearningmastery.com/k-fold-cross-validation/

[6] https://www.researchgate.net/publication/324701535_Cross-Validation

[7] https://www.sciencedirect.com/science/article/pii/S0377839818301105