Artificial Intelligence, University of Bucharest

Practical Machine Learning

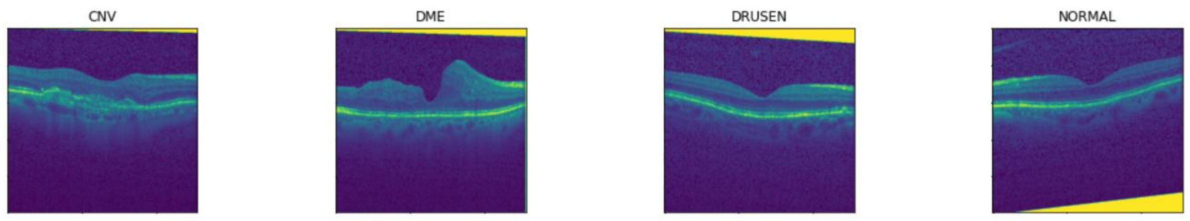
Project 2: Unsupervised learning – image clustering

Student: Sotir Anca-Nicoleta (group 407)

In this documentation will be presented my approaches to classify and cluster images from a dataset containing retinal optical coherence tomography scans from healthy people and from people suffering from one of three different diseases. A first attempt at solving this problem was establishing a supervised learning method by training a convolutional neural network, which obtained very good results. Then, to explore unsupervised methods, two clustering algorithms, namely K-means and DBSCAN, were used on top of multiple feature extraction methods, such as using an autoencoder, the pre-trained VGG16 model and others.

Dataset description

The retinal OCT images[1] dataset contains images that can be labeled into four categories by the diagnosis: choroidal neovascularization, diabetic macular edema, drusen and normal (they will be referred to as CNV, DME, DRUSEN, NORMAL). A sample image from each class can be visualized in the following figure:



The dataset contains a total of 84 thousand images, split into training and test data as such:

Dataset	CNV	DME	DRUSEN	Normal	Total
Train	37.2k	11.3k	8.6k	26.3k	**83 484**
Test	250	250	250	250	**1000**

It should also be noted that 20% (16 696) of the training data was reserved for validation.

Machine learning methods

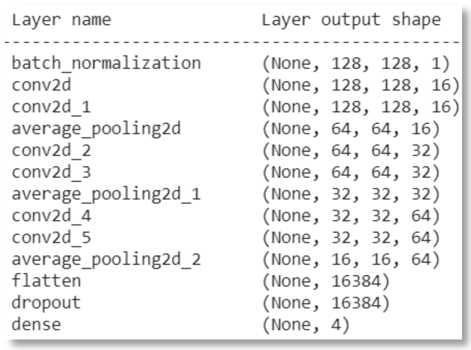
A supervised approach is a convolutional neural network trained to classify the images. The unsupervised algorithms are K-means and DBSCAN; the input for these algorithms: using pixel values, training an autoencoder and using the encoder output, histogram of oriented gradients and using the pre-trained VGG16 model (eliminating the fully connected layers).

[1] The dataset can be found **here** on Kaggle [Kermany, Daniel; Zhang, Kang; Goldbaum, Michael (2018), "Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification", Mendeley Data, V2, doi: 10.17632/rscbjbr9sj.2].

The supervised model

The images were organized into folders corresponding to the four categories and so the labels can easily be determined. This allows training a supervised model and also to determine the performance of the following unsupervised methods.

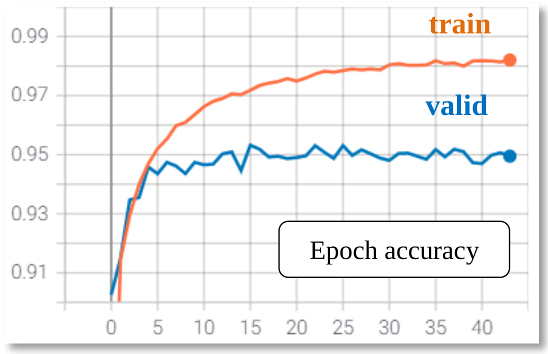
The convolutional neural network starts with a batch normalization layer (it should be noted that the images were scaled to be in the [0, 1] range beforehand). It contains three successions of two convolutional layers followed by average pooling. A dropout layer was also added before the final fully connected layer. The model was trained using Adam optimizer.

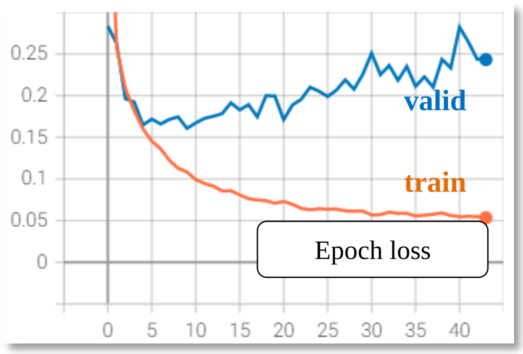


```
Layer name                  Layer output shape
--------------------------------------------------
batch_normalization         (None, 128, 128, 1)
conv2d                      (None, 128, 128, 16)
conv2d_1                    (None, 128, 128, 16)
average_pooling2d           (None, 64, 64, 16)
conv2d_2                    (None, 64, 64, 32)
conv2d_3                    (None, 64, 64, 32)
average_pooling2d_1         (None, 32, 32, 32)
conv2d_4                    (None, 32, 32, 64)
conv2d_5                    (None, 32, 32, 64)
average_pooling2d_2         (None, 16, 16, 64)
flatten                     (None, 16384)
dropout                     (None, 16384)
dense                       (None, 4)
```

Layer parameters:

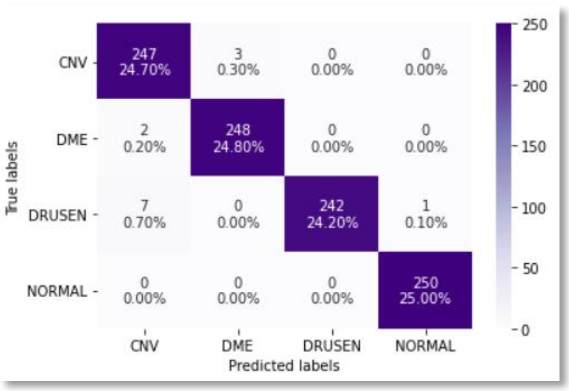
- convolutional – 3x3 kernel with He uniform initializer, ReLU activation and padding; the number of filters: 16 for the first two convolutional layers, 32 for the middle convolutional layers and 64 for the last two convolutional layers.
- average pooling – size 2, stride 2 horizontal and 2 vertical, with padding
- dropout – rate 0.4
- dense – softmax activation to use categorical crossentropy loss (output represents 4 probabilities for the image to belong to each class)





During training, the model was starting to overfit as it can be seen that the validation loss started to increase after the 5th epoch. Because of this, the model used for testing was the one corresponding to the 5th epoch. It can also be seen that the model's accuracy on the validation data is around 95%.

The model's accuracy on the test set is 98.7% and a confusion matrix was also computed.



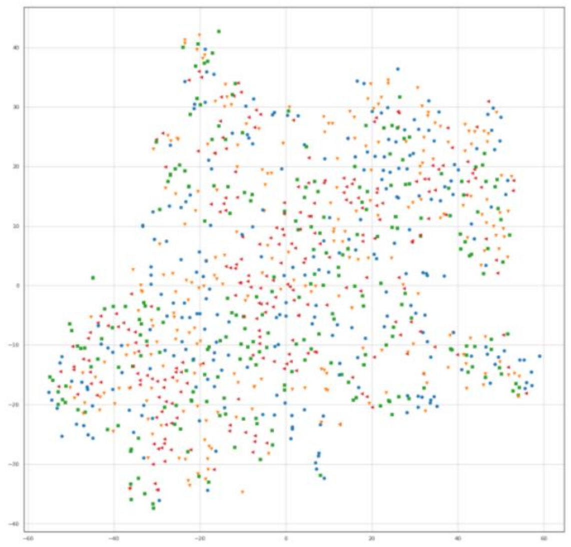
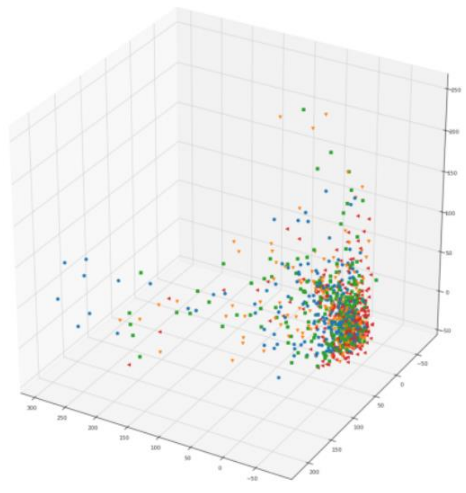
True labels \ Predicted labels	CNV	DME	DRUSEN	NORMAL
CNV	247 24.70%	3 0.30%	0 0.00%	0 0.00%
DME	2 0.20%	248 24.80%	0 0.00%	0 0.00%
DRUSEN	7 0.70%	0 0.00%	242 24.20%	1 0.10%
NORMAL	0 0.00%	0 0.00%	0 0.00%	250 25.00%

The unsupervised approach

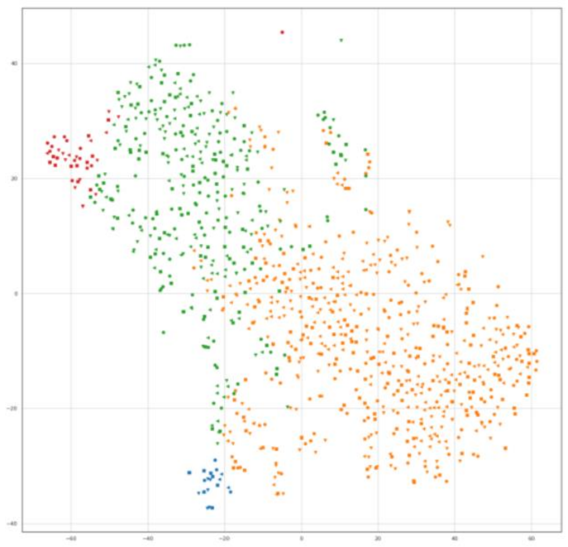
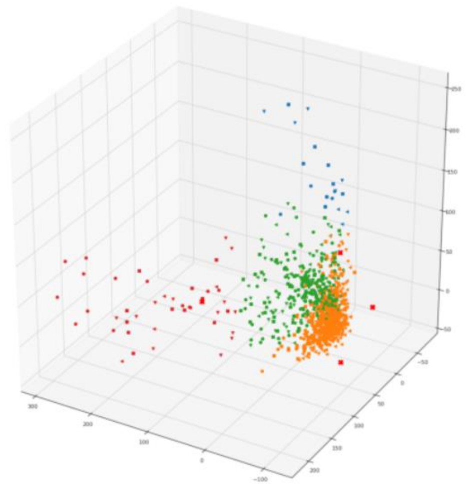
As stated before, K-means and DBSCAN were used. Because the data is represented by images, feature extraction is very important as it can impact the efficiency of the unsupervised clustering algorithms. For each method, the features can be visualized in 2D and 3D using dimensionality reduction (t-SNE and PCA). For DBSCAN, the hyperparameters are tuned using the following heuristic: plotting the ordered distances to the kth neighbor and choosing the point which has the biggest change in distance.

1. Clustering on pixel values

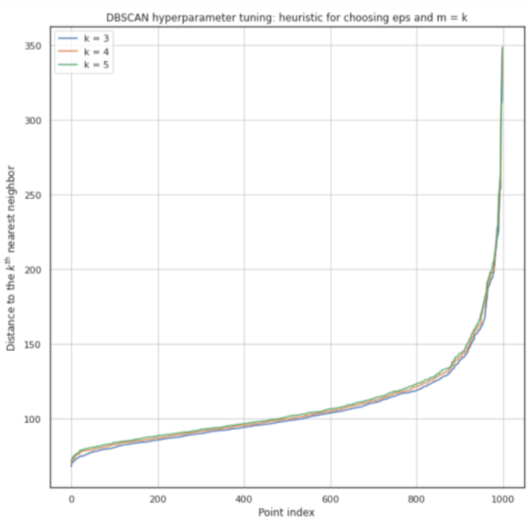
The images were scaled beforehand to be in the [0, 1] range. Visualizing the features can initially show if the features are well separated or not.

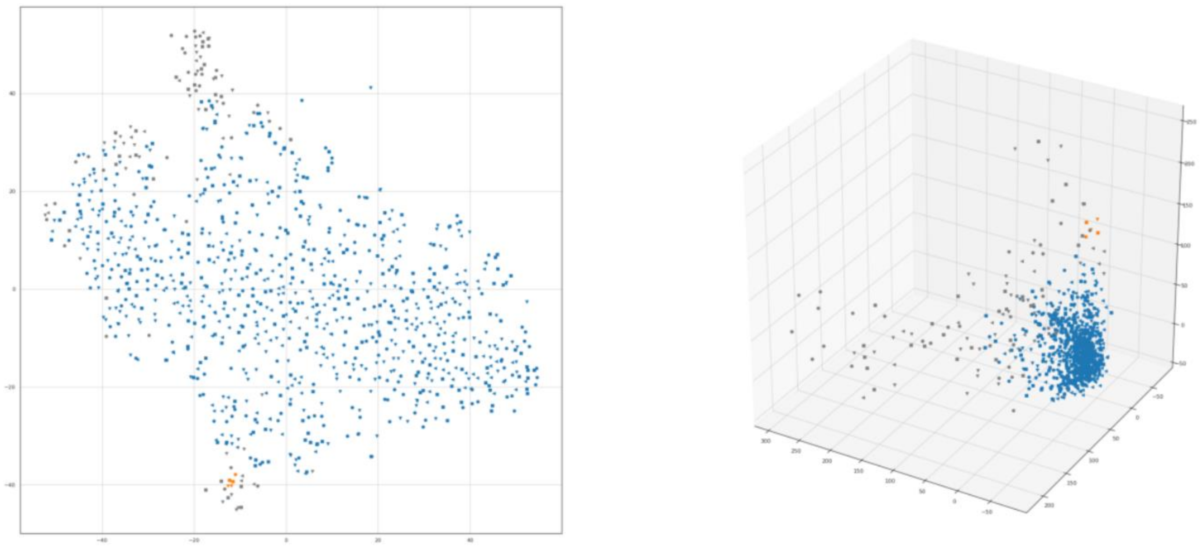
2D and 3D visualization of the pixel value features with true labels. It can be seen that the categories are mixed and an initial idea would be that the clustering will not be so successful.

The results for the K-means clustering (given parameter 4).
Accuracy 29.4% is just a little better than random chance.



The heuristic for choosing DBSCAN hyperparameters (in this case, chosen ε is 130 and m is 5).

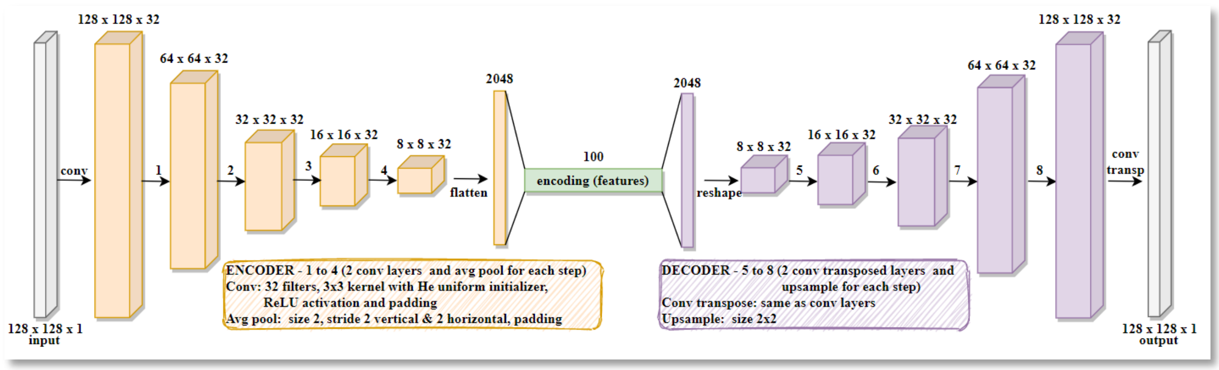


The results for the DBSCAN clustering (given parameters 130 and 5 based on the heuristic). Accuracy 23.8% is worse than random chance. It can be seen that most of the points are grouped in the same cluster based on the way data is spread.

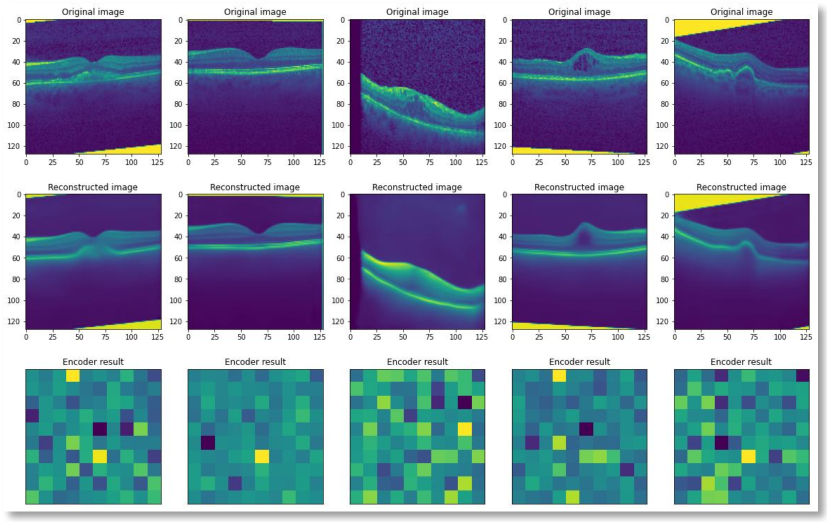
2. Training an autoencoder and clustering on the encoder output

Another idea was to use the middle layer output of an autoencoder network as the representative features for images. An autoencoder is formed of two networks – an encoder and a decoder – which are trained together. The principle is reducing the input image to a small representation that still contains relevant information (this part being accomplished by the encoder network); this small representation becomes the input of the decoder network, which aims to recreate the initial image. Thus, the loss of the autoencoder network is the mean squared error between the input image and the reconstructed image. Finally, the decoder can be discarded after training and clustering methods can be applied on the encoder's output.

It should also be noted that the autoencoder network is unsupervised, because it does not use the four categories labels.

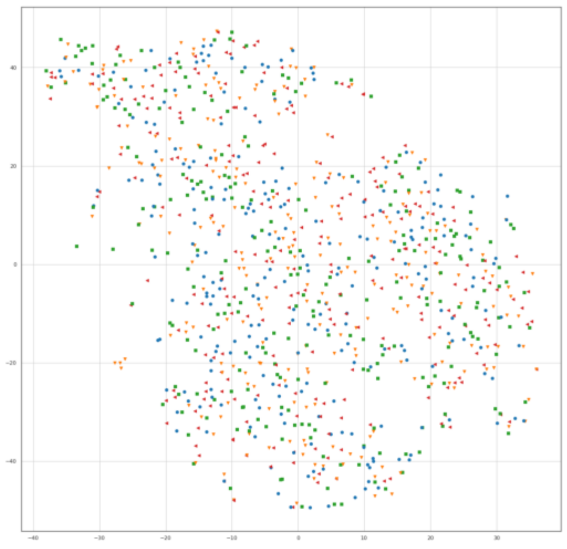

128 x 128 x 1
input
conv
128 x 128 x 32
1
64 x 64 x 32
2
32 x 32 x 32
3
16 x 16 x 32
4
8 x 8 x 32
flatten
2048
100
encoding (features)
2048
reshape
8 x 8 x 32
5
16 x 16 x 32
6
32 x 32 x 32
7
64 x 64 x 32
8
128 x 128 x 32
conv
transp
128 x 128 x 1
output
ENCODER - 1 to 4 (2 conv layers and avg pool for each step)
Conv: 32 filters, 3x3 kernel with He uniform initializer, ReLU activation and padding
Avg pool: size 2, stride 2 vertical & 2 horizontal, padding
DECODER - 5 to 8 (2 conv transposed layers and upsample for each step)
Conv transpose: same as conv layers
Upsample: size 2x2

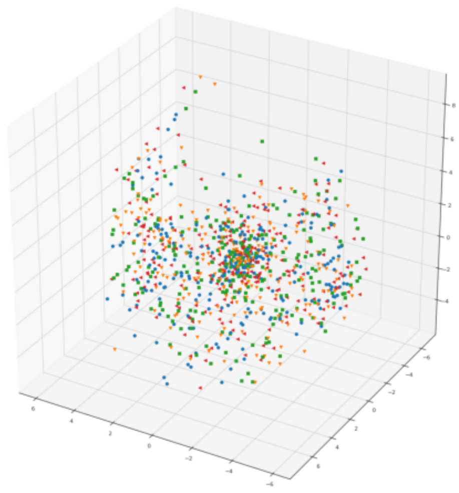
Autoencoder architecture. The autoencoder was trained for 50 epochs with Adam optimizer.



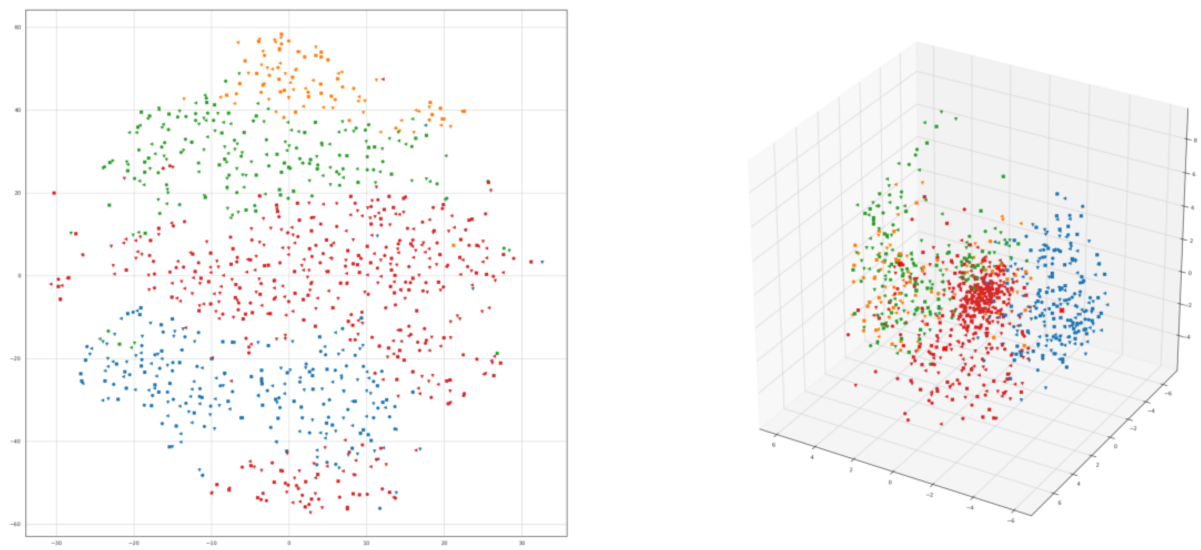
Five sample images, below each of them is the corresponding autoencoder output. The reconstructed images are much smoother, the retina's shape is preserved well. The last row contains their corresponding 10x10 features extracted with the encoder only.

After the autoencoder is trained, the middle layer output (the features, which are arrays of size 100) can be considered for clustering. Their 2D and 3D representation can be seen below:

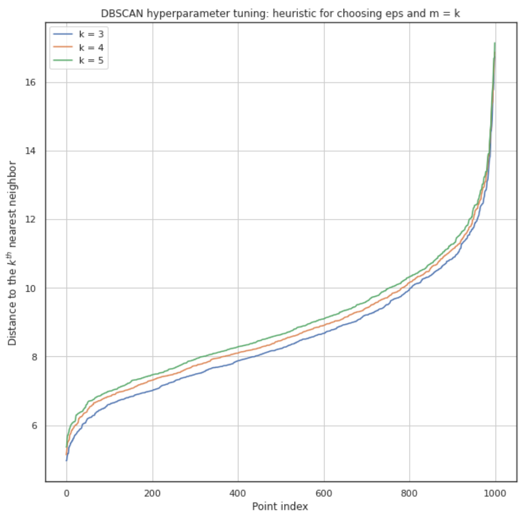




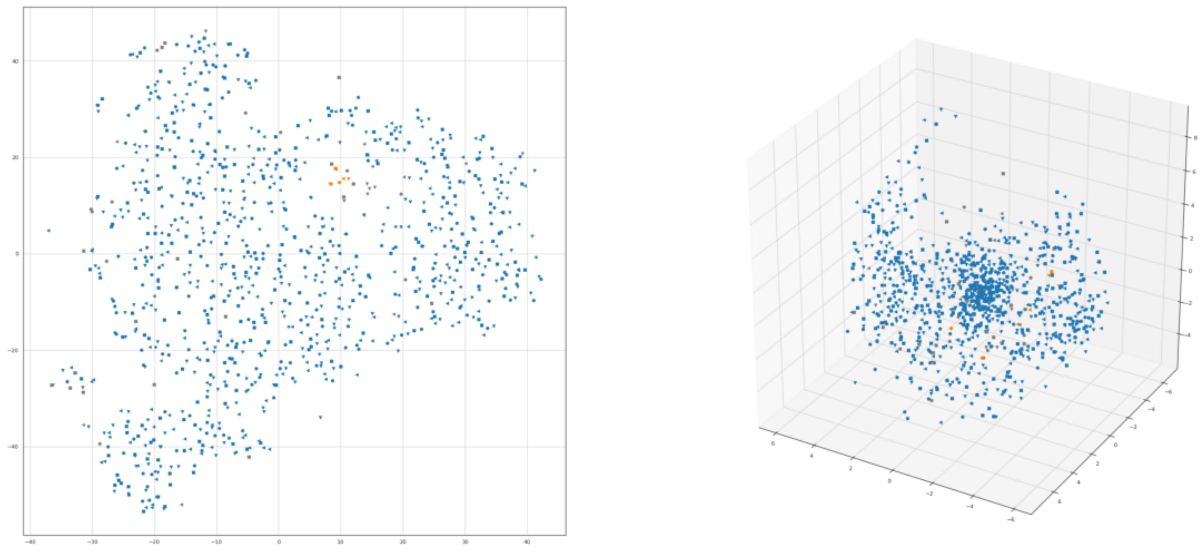
2D and 3D visualization of the encoder features with true labels. Again, it can be seen that the categories are mixed.



The results for the K-means clustering (given parameter 4).
Accuracy 25.9% is barely any better than random chance.



The heuristic for choosing DBSCAN hyperparameters (in this case, chosen ε is 11 and m is 3).

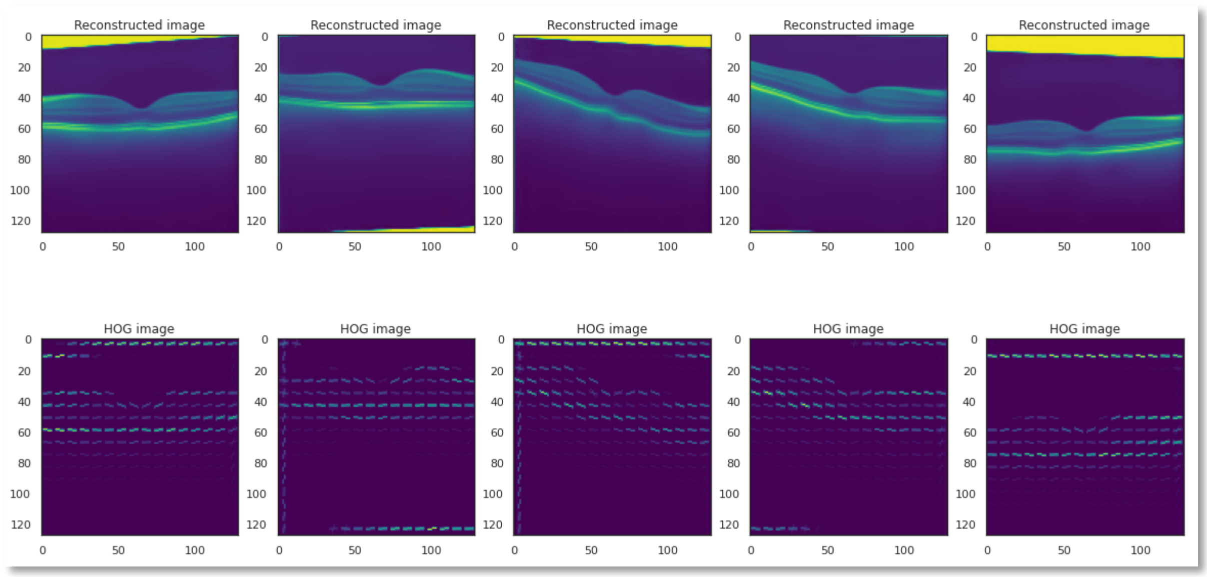


The results for the DBSCAN clustering (given parameters 11 and 3 based on the heuristic).
Accuracy 24.2% is worse than random chance. Again, most of the points are grouped in the same cluster.

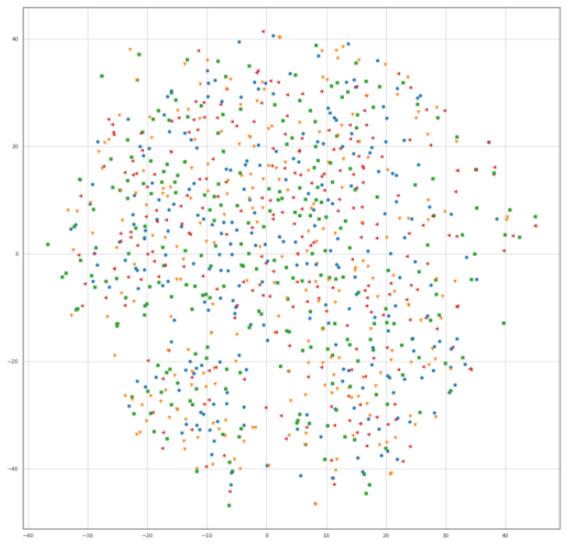
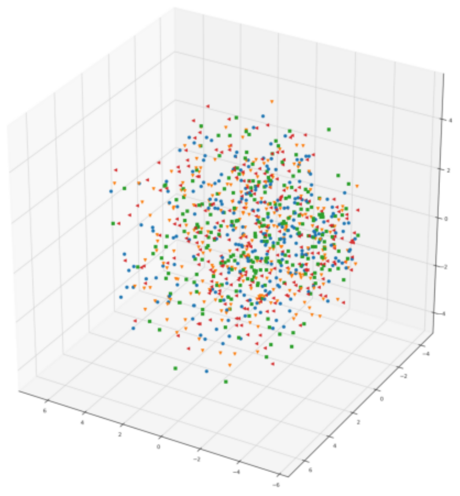
3. Clustering on histograms of oriented gradients

HOG can be very useful for tasks such as object detection. Even though they would not necessarily seem very useful in the case of retinal OCT images, an idea was that they may enhance the form of the retina and continue to show different deformities present in the three illnesses.

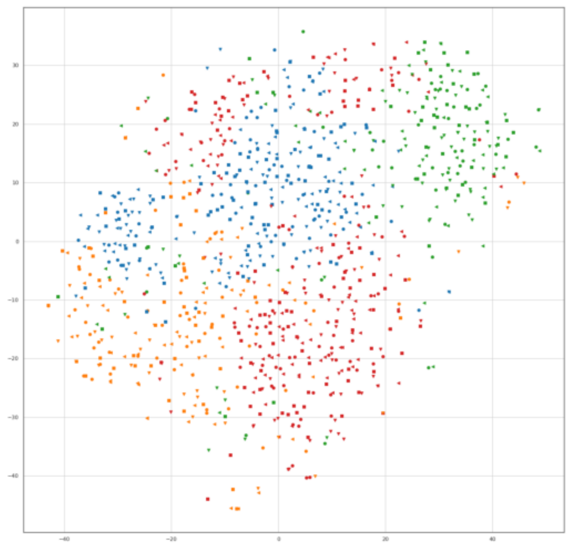
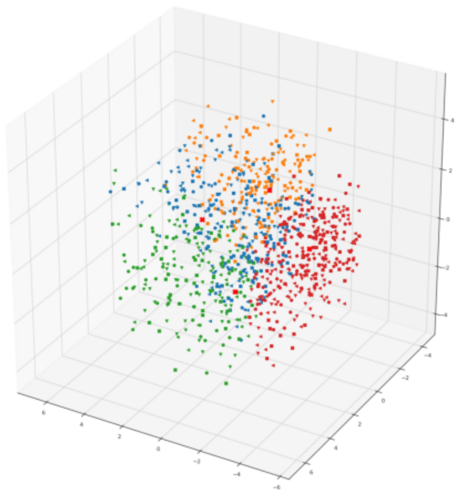
The original images are fairly noisy and so, the shape of the retina is not that well perceived in the HOG representation. Because of that, the HOG representations are computed for the autoencoder reconstructed images, which are a lot smoother and preserve the shape of the retina. It should be noted that denoising was also attempted instead of using the autoencoder, but the shape of the retina was becoming vague in that case.



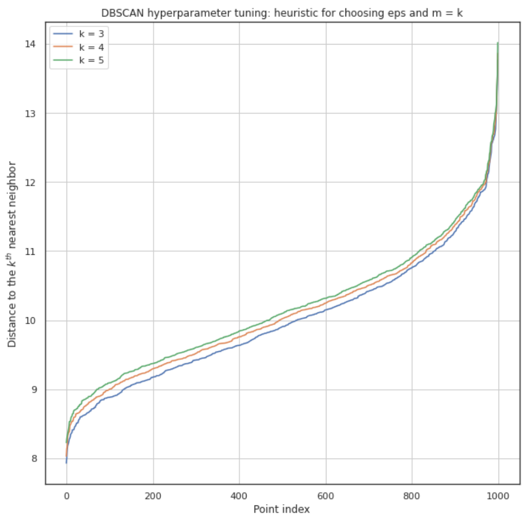
Five reconstructed images and their HOG representations (8 orientations, 8x8 pixels per cell).

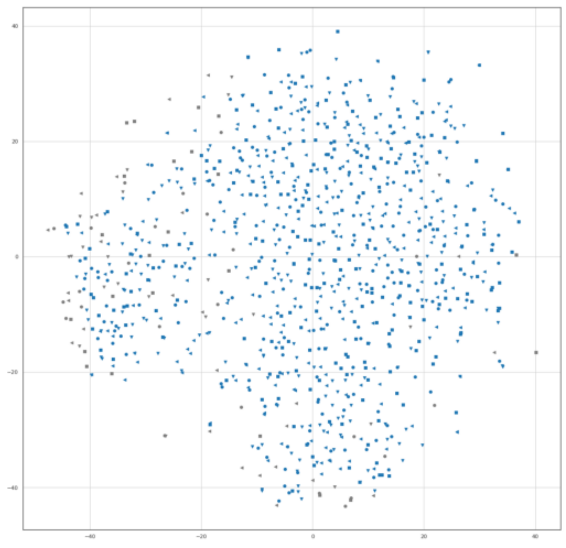
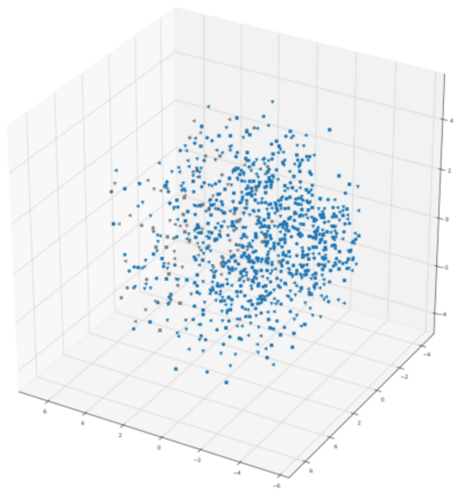
2D and 3D visualization of the HOG features with true labels.
In this case also, the categories are mixed.

The results for the K-means clustering (given parameter 4).
Accuracy 28.1% is just a little better than random chance.



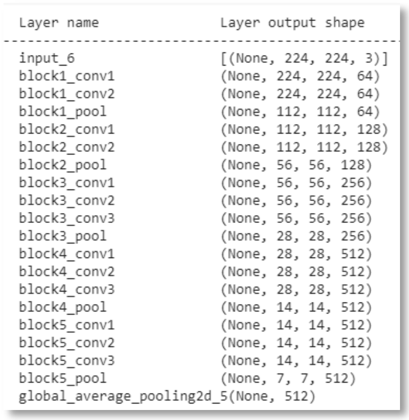
The heuristic for choosing DBSCAN hyperparameters (in this case, chosen ε is 11 and m is 3).

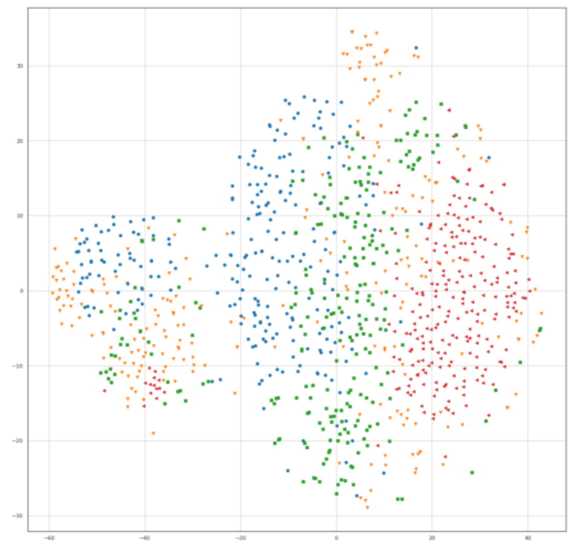
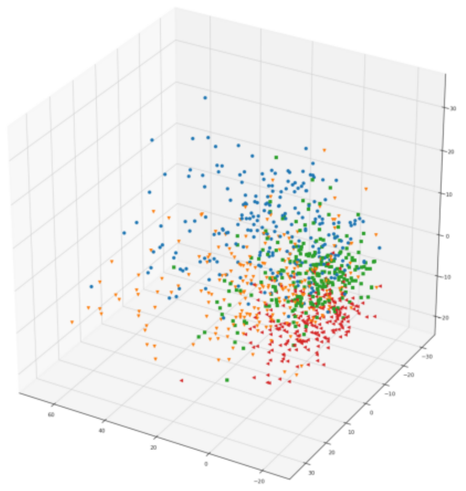
The results for the DBSCAN clustering (given parameters 11 and 3 based on the heuristic).
Accuracy 22.8% is worse than random chance. Most of the points are again grouped in the same cluster.

4. Clustering on the output of a pre-trained VGG16 model

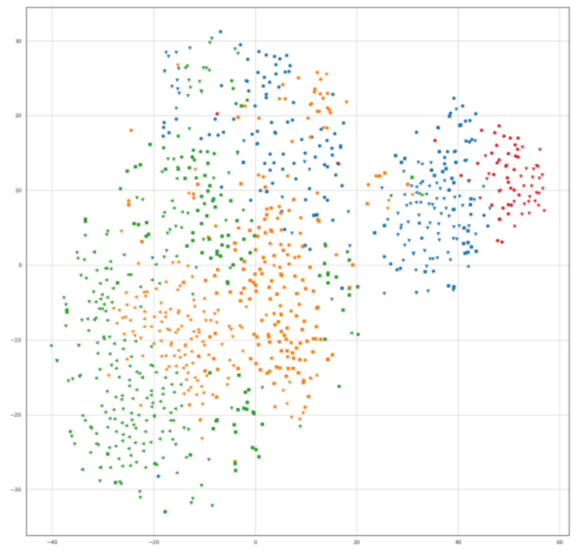
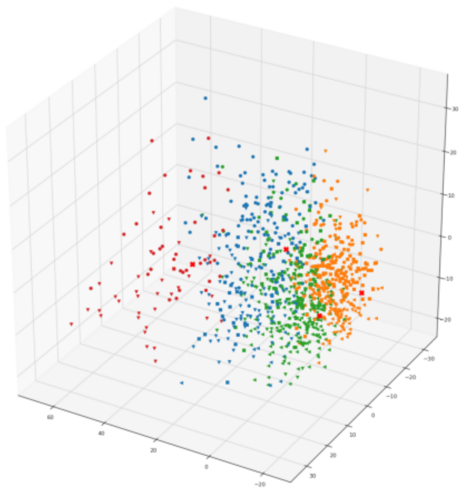
The VGG model is trained on the ImageNet dataset to classify images into one of 1000 categories. Keras provides an option to create such a model and use its pre-trained weights on different tasks (transfer learning). The ImageNet dataset, however, is very different to the retinal OCT images and so the question is how well the pre-trained VGG model would perform at this respective task. For this, the VGG16 version of the model was used by eliminating the last fully connected layers which contribute to the classification. The new model's output represents the features on which the clustering models will be applied.



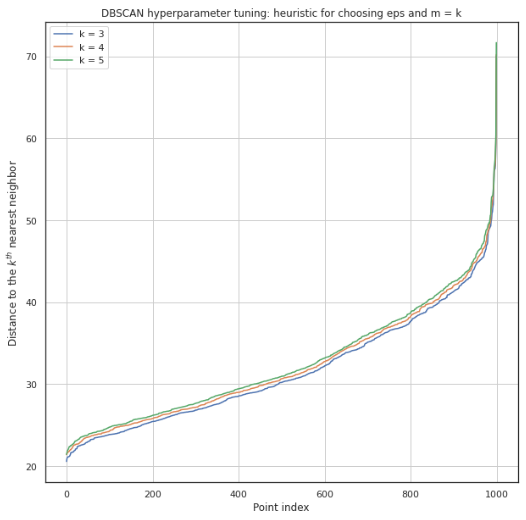
```
Layer name                 Layer output shape
------------------------------------------------
input_6                    [(None, 224, 224, 3)]
block1_conv1               (None, 224, 224, 64)
block1_conv2               (None, 224, 224, 64)
block1_pool                (None, 112, 112, 64)
block2_conv1               (None, 112, 112, 128)
block2_conv2               (None, 112, 112, 128)
block2_pool                (None, 56, 56, 128)
block3_conv1               (None, 56, 56, 256)
block3_conv2               (None, 56, 56, 256)
block3_conv3               (None, 56, 56, 256)
block3_pool                (None, 28, 28, 256)
block4_conv1               (None, 28, 28, 512)
block4_conv2               (None, 28, 28, 512)
block4_conv3               (None, 28, 28, 512)
block4_pool                (None, 14, 14, 512)
block5_conv1               (None, 14, 14, 512)
block5_conv2               (None, 14, 14, 512)
block5_conv3               (None, 14, 14, 512)
block5_pool                (None, 7, 7, 512)
global_average_pooling2d_5(None, 512)
```

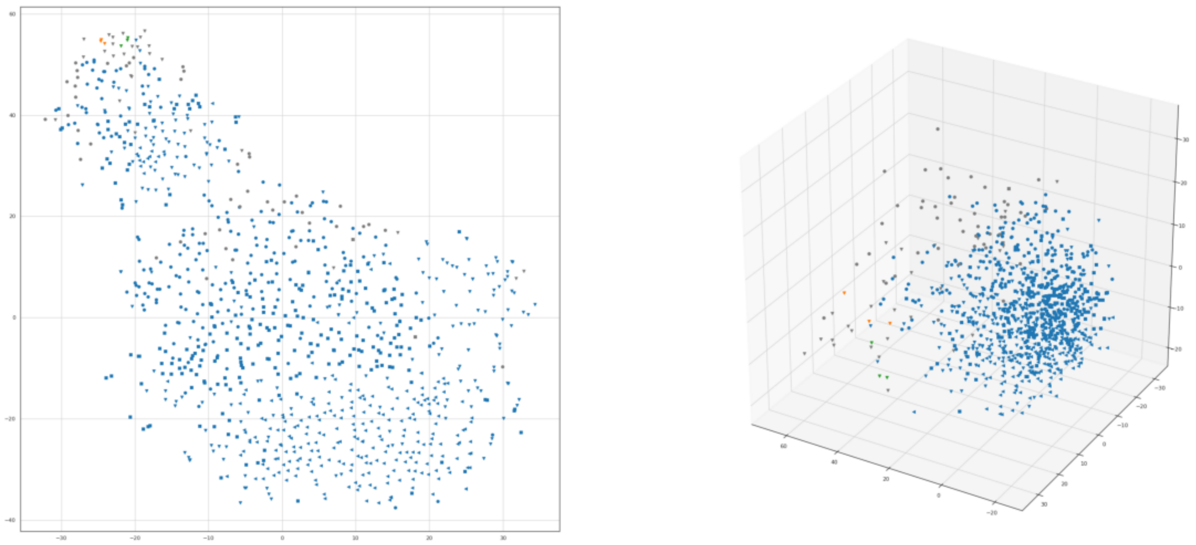
2D and 3D visualization of the VGG16 features with true labels. This is the first case in which the categories are more separated; this would lead one to believe that the clustering algorithms would perform better than before.

The results for the K-means clustering (given parameter 4). Accuracy 41.7% is considerably better than the other cases (none of which had the accuracy greater than 30%), confirming the assumption made above.



The heuristic for choosing DBSCAN hyperparameters (in this case, chosen ε is 40 and m is 3).



The results for the DBSCAN clustering (given parameters 11 and 3 based on the heuristic). Accuracy 25.3% is just as good as random chance. Again, most of the points are again grouped in the same cluster.

Final results and conclusions

The main purpose of this project was to apply and compare unsupervised methods – two clustering algorithms (K-means and DBSCAN) – however, the random chance baseline and a supervised model baseline are also considered when comparing the results.

The following table presents the scores on the test data (accuracy and normalized mutual information) for the methods described in this documentation:

Features	Method	ACC	NMI
–	**Random chance**	25%	–
	Supervised CNN	**98.7%**	–
Pixel values	**K-means**	29.4%	0.0092
	DBSCAN	23.8%	0.0086
Encoder	**K-means**	25.9%	0.0018
	DBSCAN	24.2%	0.0023
HOG	**K-means**	28.1%	0.0041
	DBSCAN	22.8%	0.0002
VGG	**K-means**	**41.7%**	0.1297
	DBSCAN	25.3%	0.0668

A first observation is that the only method which had very good results is the supervised convolutional neural network, with an accuracy score of 98.7%. For the unsupervised methods, the only one with considerably better results than random chance is using the pre-trained VGG model as a feature extractor. This is an example of supervised transfer learning, but is still taken into consideration for this project because the ImageNet dataset, on which the network is trained on, is very much different from the retinal OCT scans dataset.

As for comparing the two clustering algorithms, the main observation would be that K-means yielded better results than DBSCAN for the following reasons: the features were evenly dense, which led to DBSCAN assigning almost all entries in the same cluster. In addition, K-means always fixed four centroids (the algorithm must be provided with the number of clusters into which to assign data). DBSCAN was always just a little worse than the random chance case, while K-means had the possibility of achieving a good result (provided that the features were somewhat separable, which was the case only for VGG).

Finally, it was expected that the supervised model would have better results than any clustering method because of the fact that it is trained with awareness of the labels. That being said, there is still a lot of room for improvement as the best unsupervised method had only 41.7% accuracy, while the supervised network had very good results at 98.7% accuracy.