KU LEUVEN

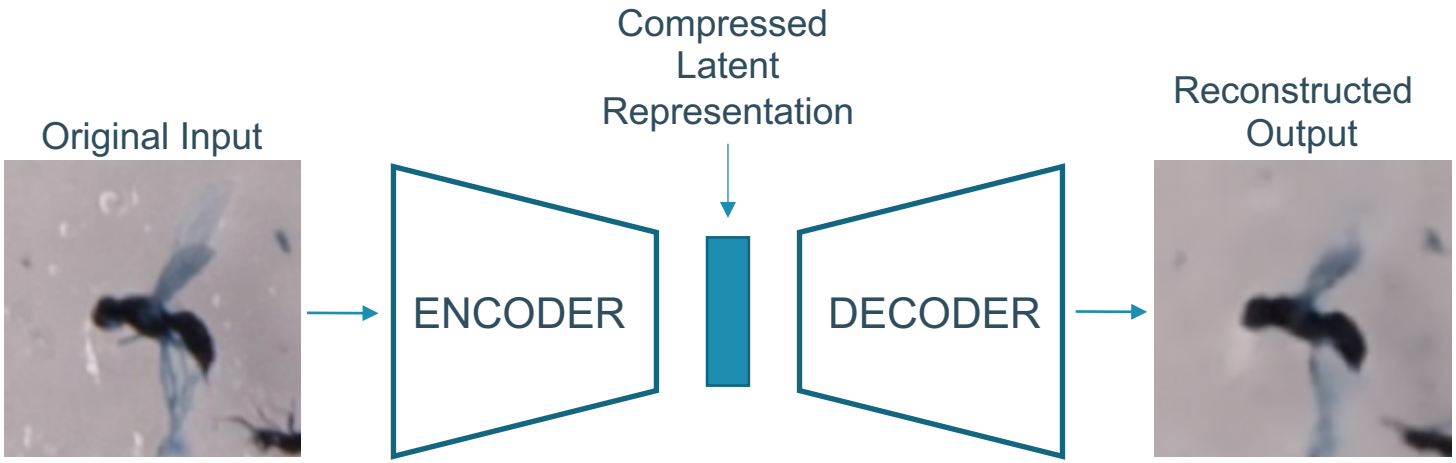
Unsupervised mislabeled image detection using Autoencoders

Application on sticky trap insect data

Abstract

- Annotated data of insects caught in sticky traps presents several labelling mistakes due to the complexity of the task. Many insect species are morphologically similar to each other and often of very small size, which makes them harder to recognize through images shot in standard cell phone cameras. Therefore, even trained entomologists might fail to accurately label all images.
- Since a cleaned and correctly annotated dataset is vital for many computer vision tasks, we explored an unsupervised machine learning methodology for identifying and removing wrong-labelled images in the dataset.
- Convolutional autoencoders generate a lower dimensional feature representation of images in an unsupervised manner. Assumingly, theses features contain the most relevant information of the images to allow for a proper reconstruction though the decoding phase. Using the DBSCAN clustering algorithm on each individual class on the encoded features results in the detection of outliers which are often either poor or mislabeled samples.
- Work inspired by: Yang, Y., & Whinston, A. (2021, November). Identifying mislabeled images in supervised learning utilizing autoencoder.

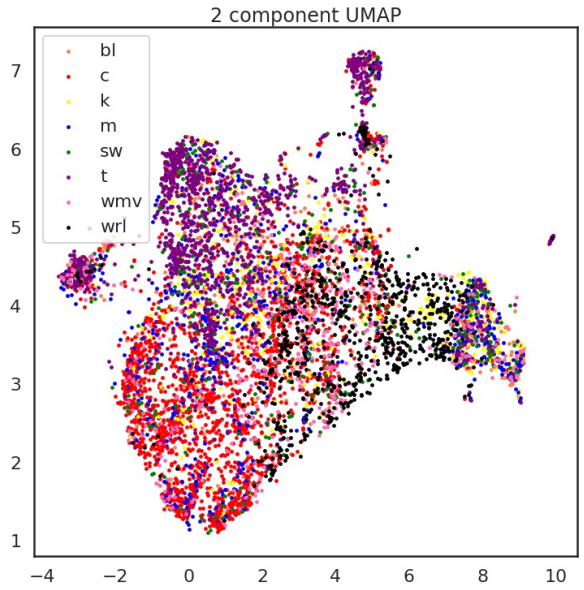
Background - Autoencoders



Since we are dealing with images, we work with convolutional autoencoders (CAEs). The encoder is a regular CNN made of convolutional and pooling layers. It reduces the spatial dimensionality of the input images while increasing the depth (i.e., filters/feature maps). The latent representation is the output of the encoder, which is also the input to the decoder. Finally, the decoder performs the inverse job of the encoder by upscaling the image back to its original dimensions. This can be achieved by the so-called transpose convolutional layers, or through upsampling layers followed by convolutional ones.

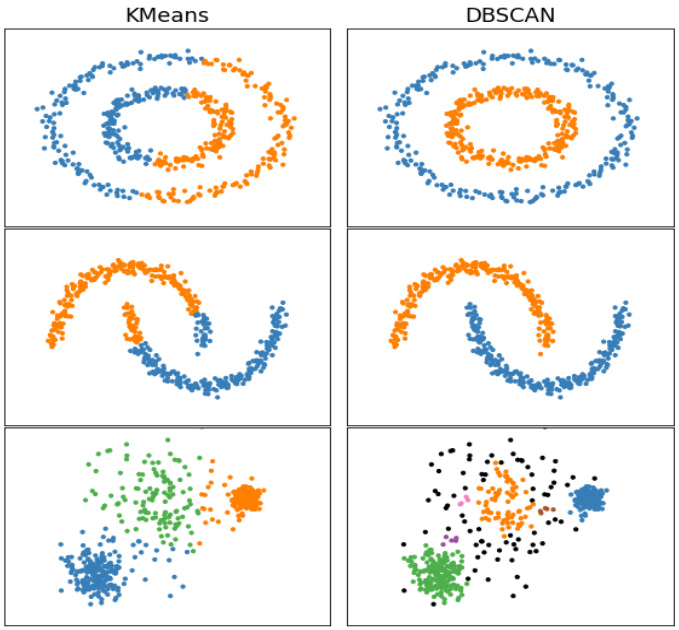
Background – DBSCAN (1)

- The latent space representation of the input images might still be high-dimensional with completely arbitrary shapes.
- Moreover, due to flawed nature of the sticky trap datasets, the classes are not perfectly separable “by eye” and there are many noise instances.
- As an illustration, we show the 2-component UMAP representation of part of the dataset. There is some class separation, but far from ideal.



Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a well-suited algorithm for this task. DBSCAN views clusters as areas of high density separated by areas of low density. Thanks to this generic view, clusters found by DBSCAN can be generically shaped as opposed to conventional clustering algorithms such as k-means, which assumes the clusters are convex shape.

Background – DBSCAN (2)



Recreated from from scikit-learn

- A DBSCAN **cluster** is a grouping of sets of core samples close to each other with sets of non-core samples that are near core samples.
- **Core samples** are points situated in areas of high-density. The concept of high-density is described by the two DBSCAN user defined parameters: ***epsilon*** and ***min_samples***. For instance, higher *min_samples* and/or lower *epsilon* result in higher density required to form a cluster

A proper choice of parameters *min_samples* and *epsilon* is crucial to perform correct clustering. These can be selected manually via simply assessing the output clusters or by some heuristic methods. In this case, *epsilon* is chosen as the value in which the slope of nearest neighbors distance plot changes more rapidly (the ”knee” of the plot). *Min_samples* is fixed as twice the dimensionality of the data. (Refer to the python notebook for more details.)

Methods

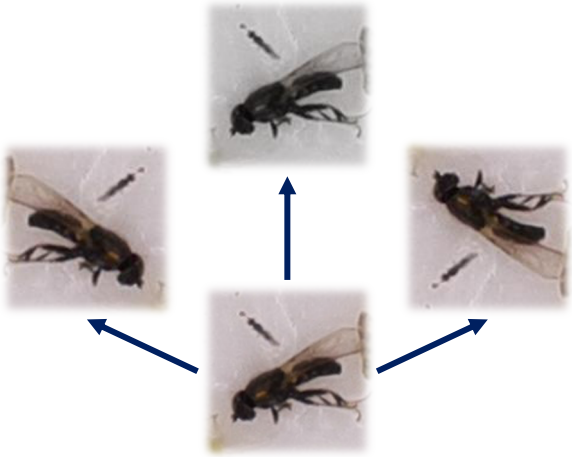
There are 2 key aspects to this methodology:

1. The appropriate choice of the CAE's architecture and successful training
2. The clustering of the latent representations (the output of the encoder) for each individual class.

Moreover, we will discuss and apply some auxiliary techniques such as data augmentation and high dimensional data visualization.

Methods - Data Augmentation

Depending on the size of your dataset, you might need to perform some data augmentation techniques. This might be especially important when training deeper architectures. Even with enough training data, it's a good technique to apply regardless, since it could help build a more robust model by making it invariant to contrast, rotation, lighting, etc.



```python
import Augmentor
img_folder = 'path_to_data'
n_samples = 2000
p = Augmentor.Pipeline(img_folder)

p.rotate(probability=0.7, max_left_rotation=12, max_right_rotation=12)
p.flip_random(probability=0.5)
p.random_color(probability=0.4, min_factor=0.2, max_factor=0.8)
p.random_contrast(probability=0.4, min_factor=0.2, max_factor=0.8)
p.sample(n_samples) #Generate augmented samples
```

Augmentor Docs.

There are many python libraries to perform augmentation. One example is *Augmentor*: all you need is to define the transformations, and the probability of them happening to each image.

Methods - CAE

Our goal is to use the latent encoded representation for outlier detection, hoping to find mislabeled samples. To that end, the encoded features must be representative of the original classes while ignoring undesired information such as background lighting or small artifacts. **The focus in this case lies on choosing an appropriate latent representation size**: large enough for good reconstruction (through the decoder) and compact enough to retain the most important information. Moreover, if the dimension of the latent representation is bigger than the number of samples in the class, DBSCAN won't work.

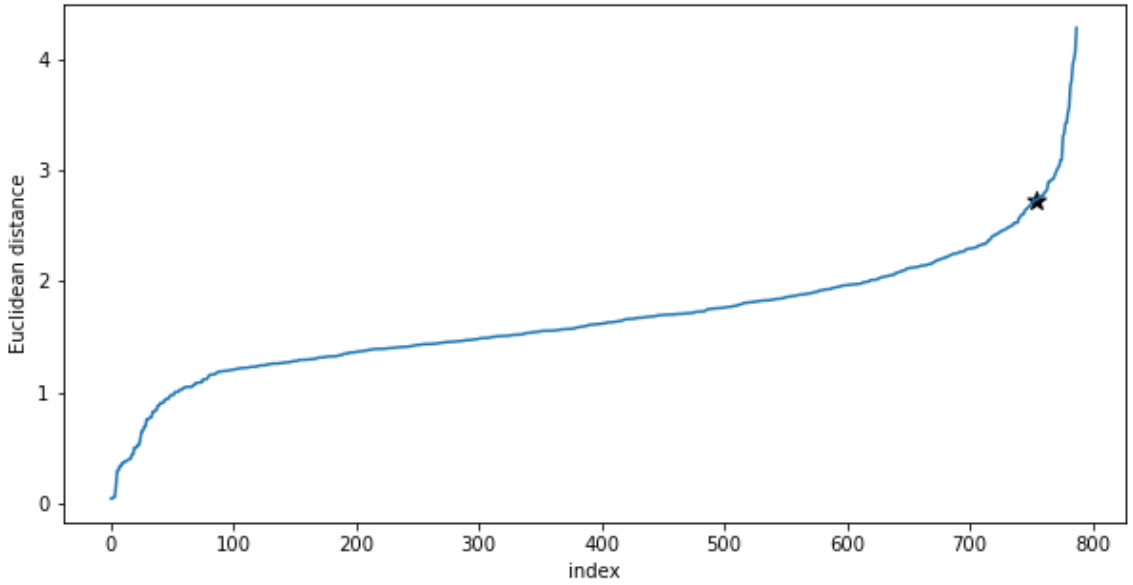
In summary, the main design choices are:

- The depth of both encoder and decoder.
- Specifics of each conv layer: number of filters, stride size…
- Other layer types: dense, dropout…
- The encoded size

For training, the Adam optimization algorithm with MSE Loss is the standard choice for simple CAEs. A validation set is a good choice to avoid overfitting, and the CAE can be assessed though a test set or by visualizing the reconstructed images.

Methods - DBSCAN

The implementation of DBSCAN on python is straight forward through the sklearn.cluster.DBSCAN package. The clustering is performed in the latent features of each class independently. The optimal value of epsilon is found at the knee of the distance plot. The distances in this plot refer to the average distance of each data point to its *min_samples* neighboring points. The outliers detected for each class are removed from the original dataset.



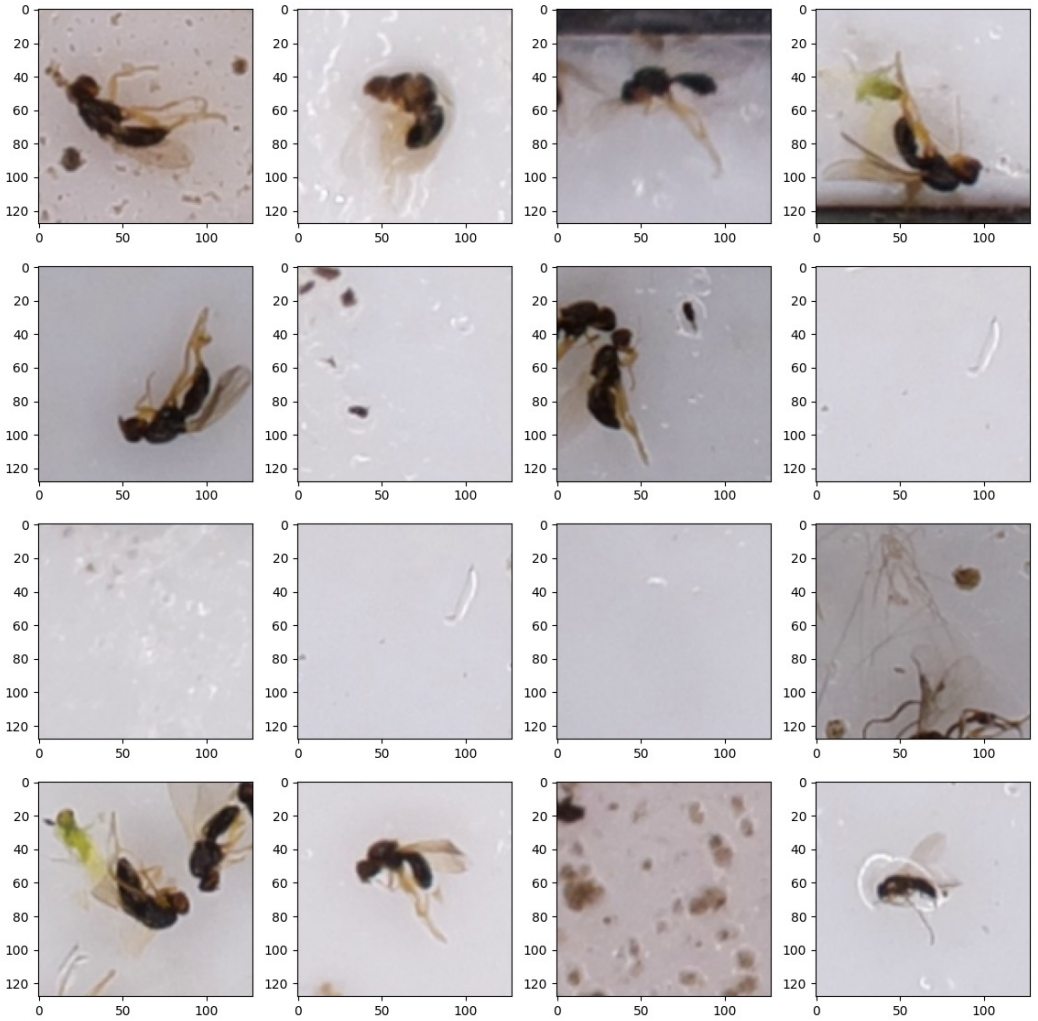
Results

Assessing the performance of this methodology is not trivial since we do not know, a priori, which samples were mislabeled. We do have an idea however for the most obvious cases. Also, there are other methods which offer some light on whether the method offers some improvements or not. Here we explore two:

1. **Visual inspection**: for the hardest cases, it's probably not wise to make assumptions that differ from the ones made by trained entomologists. There are however pictures with very bad resolution, multiple instances of insects, or even just background. These we can say for certain they are wrong or poor-quality labels.
2. **Through Machine Learning**. Training a multiclass classifier on both the original and cleaned datasets and comparing performance. The classifier and training methods must be of course identical in both cases for comparison purposes. And ideally, both trained under full cross-validation, since the datasets will be slightly different.

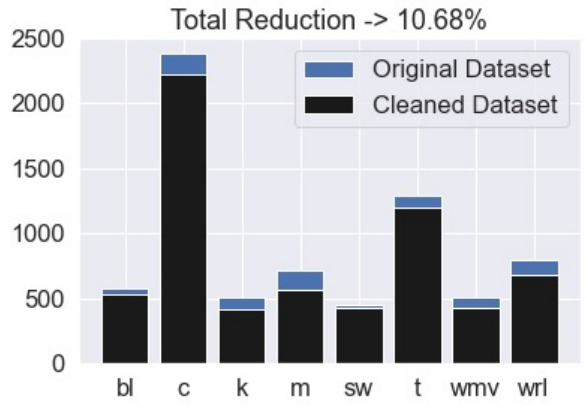
Results – Visual Inspection

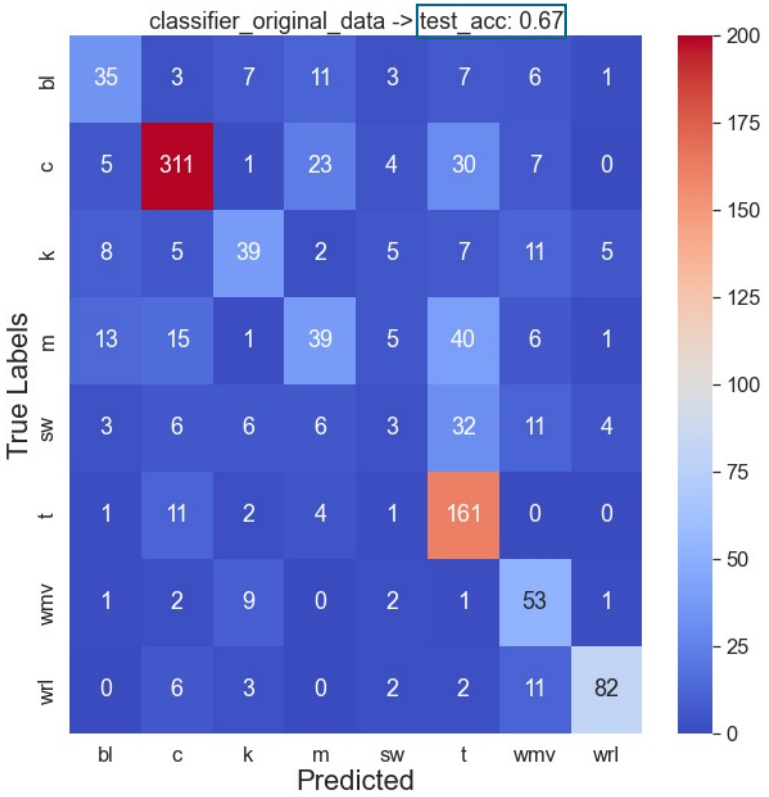
The following are some of the ‘wrl’ samples removed by a 512-sized latent representation with further PCA reduction. Some instances removed don’t look like the should’ve been. However, many background images or poor-quality ones were captured as outliers

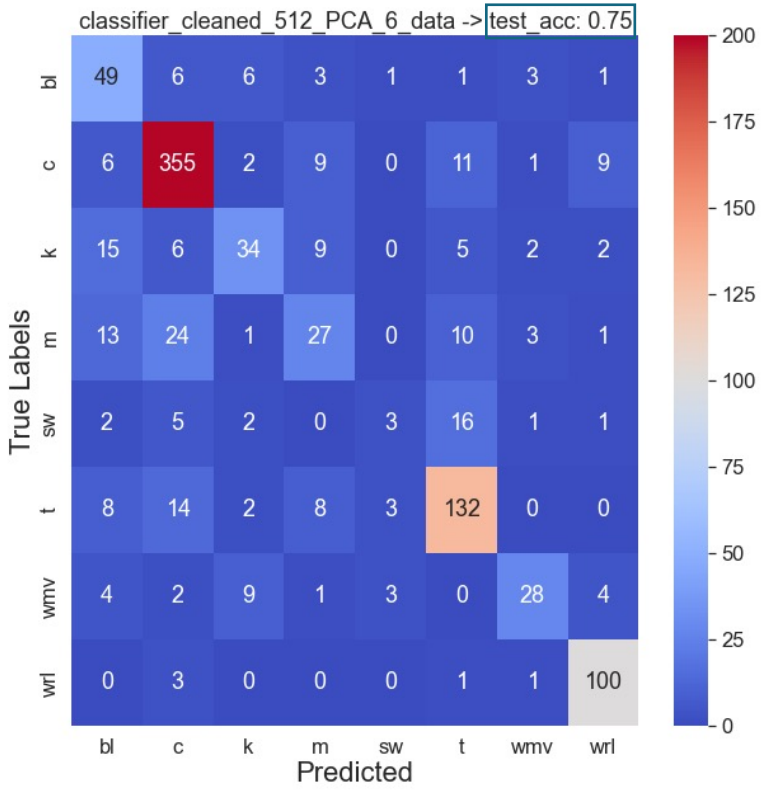


Results – Classifier

After removing the outliers, the accuracy of the model improves significantly despite the decrease on dataset size available for training.







Future Work

- Not much work on CAE architecture. A significantly better CAE could be developed with proper tuning. Other types of encoding techniques are yet to be explored.
- Testing alternatives to DBSCAN.
- Combination of dimensionality reduction techniques. Applying PCA over a large encoded latent representation offered nice preliminary results.
- Increasing the complexity of the pipeline: overlapping several outlier removal stages could improve performance. Also, training a classifier on clean and removed images.