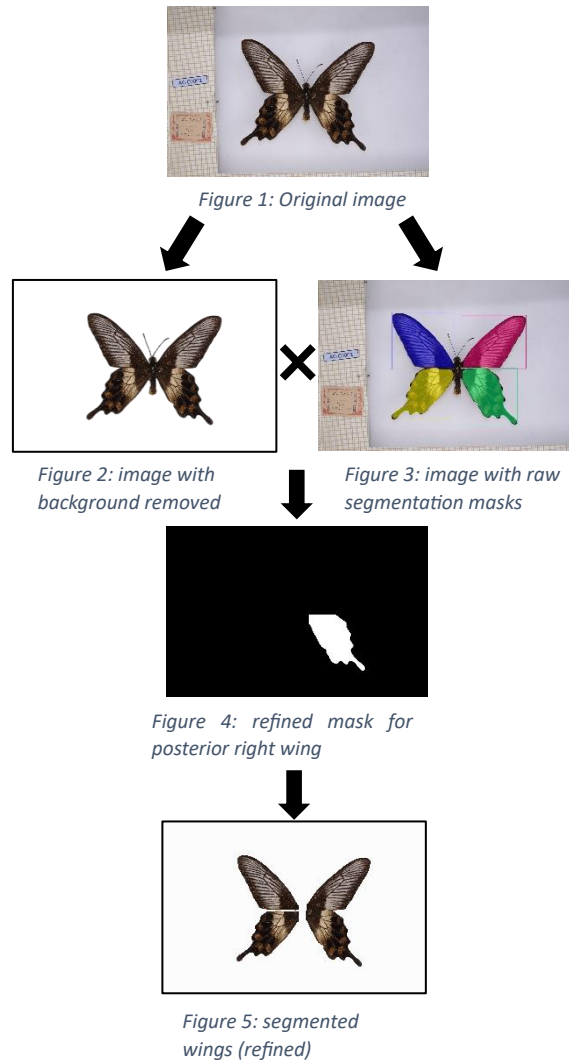


A. Segmentation

The segmentation code here is specialize to segment butterfly wings only (apart from the thorax and abdomen parts which are considered to be in the background). There are two main steps to the segmentation algorithm: (1) Segmentation of the wings based on a trained Mask-RCNN segmentation model, which is a machine learning based method that identifies pixel from each wing. This segmentation is not perfect, especially around the border of the wings, and thus there is step (2), a refinement step in which the background of the entire image is removed using the rembg package, leaving the wings, body and antennae, and then the resulting image is combined with the raw segmentation masks from the machine learning to produce refined masks with a clean border of the wings.



B. Clustering

This step creates cluster of similar phenotypes from another ImageNet trained neural network. These clusters are used for two reasons: (1) to balance the dataset if some phenotypes are over or underrepresented. (2) to later evaluate classification into these clusters from the embeddings obtained from the unsupervised training and thus have an idea of embedding space quality regarding these clusters.

If you wish to rely on your own self-made clusters, you should create a clustering folder within which there is a subfolder for each cluster containing corresponding pictures, and skip “1-VGG16_clustering”, as such (folders/images names are not important):

clustering_folder

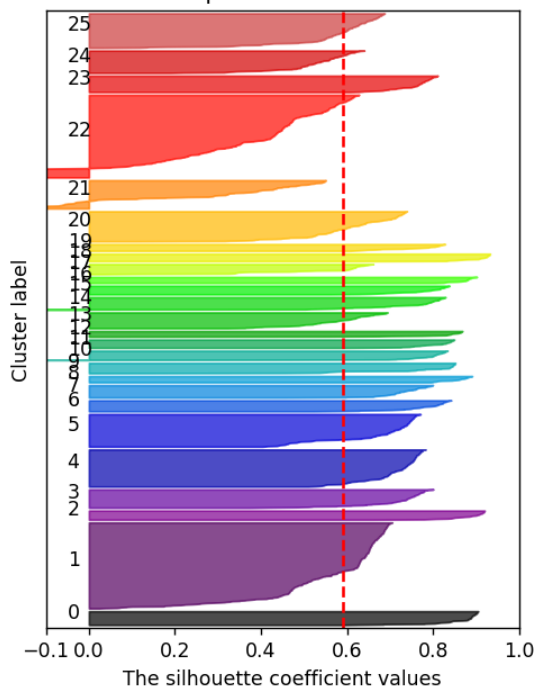
```
|__cluster_0
    |__image_from_group_0
    |__ ...
|__cluster_1
    |__image_from_group_0
    |__ ...
|__ ...
```

When running “1-VGG16-Clustering”, first all picture are run through a pretrained VGG16 network, and then the resulting embeddings are projected onto 3 dimension using a UMAP projection. For the HDBScan clustering, you will be asked to choose two parameters (check HDBScan documentation for more information). Then the 3D UMAP projection will be plotted along with clustering results as a colored scatterplot, as well as the clusters silhouette scores. Close the window, and then you can either keep this clustering result (type 1) or try with new parameters (type 0).

(Code for the plots was inspired from this page: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)

Silhouette analysis with $n_clusters = 26$

The silhouette plot for the various clusters.



The visualization of the clustered data.

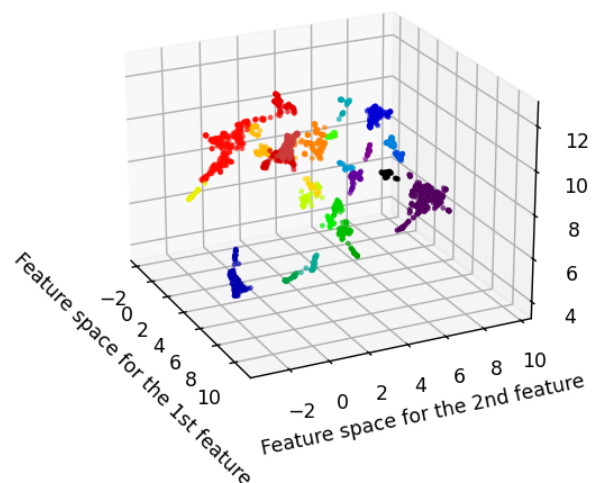


Figure 6: Silhouette and 3D scatter plot of pretrained-VGG16 image embeddings

```

Choose min_cluster_size: 20
Choose cluster_selection_epsilon: 0.1
Number of clusters: 25
Number of outliers: 599
For n_clusters = 26 The average silhouette_score is : 0.5898127

```

Figure 7: Terminal output for the clustering

After clustering, “2-create_dataset” uses these folders to create a dataset folder within which you will find: the training set that will be used to train SimCLR, which contains all of your picture data, in the folder “unlabeled” ; the training set that will be used to train the classifier for evaluation of the model, containing 80% of your data, in the folder “train” ; the validation set that will be used to train the classifier for evaluation of the model, containing 20% of your data, in the folder “validation”.

dataset_for_SimCLR_training_and_evaluation

```

|__ unlabeled
    |__ cluster_0
        |__ images
    |__ cluster_1
        |__ images
    |__ ...
|__ train
    |__ cluster_0
        |__ images
    |__ cluster_1
        |__ images
    |__ ...
|__ validation
    |__ cluster_0
        |__ images
    |__ cluster_1
        |__ images
    |__ ...

```

C. Training

To change the hyperparameters of the network, change the lines 47 to 51 of “1-simCLR_training” (bold font values):

#To use pre-determined parameter values

if param == -1 :

batch_size=**256**

max_epochs = **300**

temp = **0.5**

Or use the `param_i` argument (value 1 to 27) to explore the parameter grid, which is made of combinations of the following parameters:

```
batch_size = [64, 128, 256]
max_epochs = [50,100,300]
temp = [0.1, 0.5, 0.8]
```

To find out more about those hyperparameters and their recommended values, I recommend to read the SimCLR article: *“Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020, November). A simple framework for contrastive learning of visual representations. In International conference on machine learning (pp. 1597-1607). PMLR. »*

After training, the trained model will be saved as:

`simclr_bs_[parameter]_nepochs_[parameter]_temp_[parameter].ckpt`

D. Embed

From the saved trained model, you can generate embeddings for your images using “1-generate_embeddings”. This code will save two files: (1) a csv file containing raw 2048-dimensional coordinates, with images in rows, and each dimension in columns, and (2) a csv file with PCA projection result, keeping axes that explain more than 0.5% of the total variance, still with images in rows and dimensions in columns.