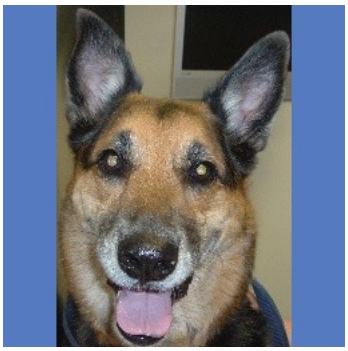
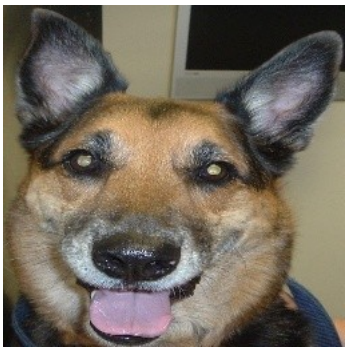
Workshop *Learning from Data*

Session 2: Dimension Reduction

BACKGROUND:

Beginning with the images we created in our last session, either

padded with a random color border or stretched and then resized, we can now apply some techniques to reduce the dimensionality of our data.

First, we might try **Principal Component Analysis (PCA)**. Scikit-Learn's basic implementation uses singular value decomposition and takes one parameter, n_components to keep, which can either be an integer (the absolute number of components) or a float between 0 and 1. In case of a float, Scikit-Learn will keep the minimum number of components that retains that percentage of the variance in the original data.

Here is example output as n_components is decreased from 99% to 75%. These exact numbers are based on particular standardization choices (from last session) and your exact results will vary.

```
(cv) workshop@vitalstatistix:~/preprocessing$ python3 preprocess.py
Loading the images from  /data/WP/WilkesPets/cropped128/
Loaded  2000 images
with n_components = 0.99  PCA reduced dimensions from 49152 to 1418
with n_components = 0.98  PCA reduced dimensions from 49152 to 1125
with n_components = 0.97  PCA reduced dimensions from 49152 to 920
with n_components = 0.96  PCA reduced dimensions from 49152 to 763
with n_components = 0.95  PCA reduced dimensions from 49152 to 638
with n_components = 0.85  PCA reduced dimensions from 49152 to 137
with n_components = 0.80  PCA reduced dimensions from 49152 to 76
with n_components = 0.75  PCA reduced dimensions from 49152 to 46
(cv) workshop@vitalstatistix:~/preprocessing$
```

Another very simple way to reduce the dimensionality of these images is to recast them as a **histogram**. Similar to a bar chart, this is the aggregation of the data values into a number of groups – the histogram counts the number of values falling into each group. For an image, this means the spatial location of the values is lost as we reduce dimensionality to the number of groups we create. Starting with the three distinct RGB channels in the image, we might produce three histograms and group each pixel three times (each color channel counting in its own histogram) or we might combine the RGB values and place it in only one group.

Color histograms are well discussed in image processing texts as they form the basis for many useful image transformations. We suggest that you follow the approach used by O. Chapelle, P. Haffner, and V. N. Vapnik in Support vector machines for histogram-based image classification from *IEEE Transactions on Neural Networks*, 10(5):1055–1064, Sept 1999.[1]

When constructing histograms, the salient parameter to vary is the number of groups (bins).

Also note, it is best to compute a histogram from the highest resolution data available. Likewise, there is no need to square images by padding or stretching. These 'standardization steps' resample and otherwise alter the original color distribution. Normalization of the histogram values (so the sum of all groups is one) allows images of differing resolutions and aspect ratios to be considered together.

TO TRY:

The program **pcaexample.py** is set to try PCA values of [0.99, 0.98, 0.97, 0.96, 0.95, 0.85, 0.80, 0.75] on files in the *SourceDirectory*, as defined near the top of the code. Running it as supplied should produce results similar to those above. If you change that variable to the *TargetDirectory* you used in **mkcroppedjpg.py**, you can run it to see how PCA reduces the dimensionality of your data.

A second program, **decisiontree.py**, uses KFold cross validation to create and test decision tree models that attempt to predict the species of each animal. Sample output is given below:

```
(cv) workshop@vitalstatistix:~/2$ python3 decisiontree.py
Loading the images from  /data/WP/WilkesPets/stretched128/
Loaded  2000 images
Performing PCA with n_components = 0.95 reduced dimensions from 49152 to 673
Running the Decision Tree classifier, maximum depth = 4
Split  1  Result:
[[175  51   0   0]
 [ 46 123   0   0]
 [  1   3   0   0]
 [  0   1   0   0]] 75.44303797468355
```

[1]This paper presents successful image classification results from SVM and kNN learners using histogram data constructed from images represented as HSV and RGB values, finding that the color space choice had minimal impact on performance because after grouping into bins, no color space information is used by the classifiers.

```
Split  2  Result:
[[177  58   0   0]
 [ 46 116   0   0]
 [  1   0   0   0]
 [  0   2   0   0]] 73.80352644836272

Split  3  Result:
[[194  48   0]
 [ 31 126   0]
 [  0   1   0]] 80.20050125313283

Split  4  Result:
[[164  54   0   0]
 [ 44 136   0   0]
 [  0   1   0   0]
 [  1   0   0   0]] 75.37688442211056

Split  5  Result:
[[190  28   0]
 [ 64 117   0]
 [  1   0   0]] 76.94235588972431

Aggregate results:
[75.44303797 73.80352645 80.20050125 75.37688442 76.94235589]
Accuracy = 76.35326119760279 +/- 2.1647749913515235
```

Note: because the dataset is imbalanced (with cats and dogs being much better represented than the other species), our experimental confusion matrices here vary in size when, at the validating stage of the KFold algorithm, some classes were not presented to the model.

The accuracy here beats the commercial platform Clarifai, but isn't up to Google or Amazon's success.

Try changing the *SourceDirectory* variable to point to your data. Might your data preparation choices produce better results? Other variables you might want to change are near the top of the code:

```
# PCA Choices
DoPCA  = True # Set to True to do PCA
PCAPer = 0.95 # Set to PCA percentage

# Decision Tree
TreeDepth = 4     # Maximum depth of decision tree
PrintText = False # Output Tree to screen as text
SaveTree  = False # Output Tree to file as .dot graphic
                  # see https://graphviz.org/documentation/
```

We'll start our next video segment with a discussion of decision trees.