Workshop *Learning from Data*

Session 1: Standardizing Image Size

BACKGROUND:

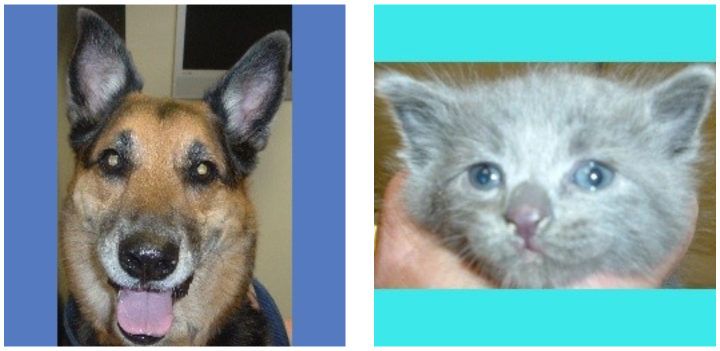
All of the images in the WilkesPets dataset have bounding boxes identifying animal faces within their metadata. A good first step is to crop the full image to this smaller region.

Some of these boxes have portrait orientation, but some are landscape.

Classifiers generally benefit from (or even require) a similar representation for all inputs.

To convert the images into a standard form, without losing data, first we will square the bounding box. To avoid losing pertinent data, we've decided to 'square' the image by extending the shorter side. For example, an image that is 100 x 120 pixels will be enlarged to 120 x 120, but an image that started 180 x 140 will become 180 x 180.

There are two obvious methods available: padding and stretching. We've supplied two python functions – **SquarePadFace** and **SquareStretchFace** that perform this operation. This gives us our first opportunity for experimentation – which method builds a better classifier for our specific task?


Images padded with a random color border

Stretched images

(As an aside, these two examples were picked for illustration because of their extreme aspect ratios. This might suggest classifying cats and dogs is as simple as looking at their aspect ratio and indeed, an experiment can be done using bounding box data as the sole input to the classifier. Such an experiment will quickly show an image's aspect ratio is not predictive.)

Merely squaring the image is not enough to standardize our data: the images still remain at different sizes. To make them all identical we can use the OpenCV function **resize**. Many machine learning algorithms operate more efficiently on matrices that are a power of 2 (e.g., 64x64, 128x128, 256x256) and some even require it. When we make the image smaller, we've done dimension reduction and throw information away. This gives us a second factor we could vary in our experiments: image size. Finally, resize also re samples the image according to an algorithm. We'll use the default, however we could vary that as well, to involve a third factor.

TO TRY:

The python program **mkcroppedjpg.py** will preprocess the data set to create cropped JPEG images ready to be submitted to a classifier.

You may edit that program to set constants near the top: set one of 'Padding' or 'Stretching' to True and set Size to a multiple of 2. The initial settings are 'Padding = True' and 'Size = 128'. Additionally there is a *TargetDirectory* variable, set to 'padded128'.

Then run the program by typing: ***python3 mkcroppedjpg.py***   If you get OpenCV errors, make sure you've already run ***workon cv*** to set the environmental paths.

Each time you run mkcroppedjpg.py, it will overwrite any previous images in the target directory (as set initially, 'padded128'). You may play with the *TargetDirectory* variable to keep multiple data sets, but you'll have to change the *SourceDirectory* variable in future programs to point to the subdirectory you want to use. Use a descriptive naming system.

Now you've readied the data to present to some classifiers!

NOTE ON FEATURE SCALING:

Another important transformation you may need to apply to your data *is feature scaling*. Feature scaling is a method used to normalize the range of data features. It is also known as *data normalization* and is usually performed during the data preprocessing step. It is recommended when numerical features of your data vectors have different scales. As our images are 8 bit per channel RGB values, data normalization isn't needed here, but if it was there are two ways to do it:

1. *Standardization* rescales the feature values so that they have a zero mean and unit variance.

2. *Min-max scaling* shifts the feature values to (usually) be within the range from 0 to 1.

Below are two Scikit-Learn functions that can help you with feature scaling:

StandardScaler:
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

MinMaxScaler:
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html?highlight=min%20max%20scaler#sklearn.preprocessing.MinMaxScaler