Welcome to Taï National Park, data scientists! In this challenge, your goal is to classify the species that appear in camera trap images collected by our research partners at the Wild Chimpanzee Foundation and the Max Planck Institute for Evolutionary Anthropology. As mentioned in the [about](https://www.drivendata.org/competitions/87/competition-image-classification-wildlife-conservation/page/411/) page, [camera traps](https://www.worldwildlife.org/initiatives/camera-traps) are one of the best tools available to study and monitor wildlife populations, and the enormous amounts of data they provide can be used to track different species for conservation efforts—once they are processed.

Thanks to our partners, we have a trove of images from camera traps located in different sites around Taï National Park. There are seven types of critters captured in this set of images: birds, civets, duikers, hogs, leopards, other monkeys, and rodents. There are also images that contain no animals. Your job is to build a model that can help researchers predict whether an image contains one of these seven types of species. Let's predict!

Tips for working with images as features

Working with images as features can be an exciting challenge. Here are some tips to consider as you begin:

GENERALIZING TO NEW SITES

As noted [above](https://www.drivendata.org/competitions/87/competition-image-classification-wildlife-conservation/page/483/#features_list), a model may erroneously learn to predict a species class based on characteristics of the environment, rather than characteristics of the species itself.

In order to make sure that our models predict well in new contexts, it is important that train and test sets have entirely different environments. In this case, we ensure this by making sure that sites are entirely in the train set or the test set, but never in both. We recommend that you take a similar approach in setting up your own splits of the training set.

Further, there are many differences to account for for images taken even within the same site. For example, consider these different images of our leopard friends:

|  |  |
| --- | --- |
| Here is an image of a leopard on the prowl (ZJ000102): | and here is an image of a leopard hiding out (ZJ000090). |
| https://drivendata-public-assets.s3.amazonaws.com/zjff-ZJ000102.jpg | https://drivendata-public-assets.s3.amazonaws.com/zjff-ZJ000090.jpg |
|  |  |
| Here is an image of a leopard investigating the camera (ZJ000097), | and here is an image of the leopard waving goodbye (ZJ000253). |
| https://drivendata-public-assets.s3.amazonaws.com/zjff-ZJ000097.jpghttps://drivendata-public-assets.s3.amazonaws.com/zjff-ZJ000253.jpg |  |

As you can see, there are significant differences in our images of leopards. The animals may be close or far from the camera, in the sun or in the shadows, or facing toward or away from the lens, among other variations. There are also differences in the color of the image, the weather conditions it was taken, and the type of camera.

In order to teach the model that these images are all of leopards despite these differences, it can be helpful to perform a preprocessing step called image augmentation. Image augmentation involves transforming the training set in multiple ways—rotating, distorting in color or sharpness, zooming in or out, are a few examples. These manipulations of the image can help the model make correct predictions in contexts it has not been exposed to before.

## **Labels**

There are eight possible labels for every image. If the image does not contain any animals, it is labeled as blank. Otherwise it must be labeled as containing one of the seven species groups included in our dataset:

1. antelope\_duiker
2. bird
3. civet\_genet
4. hog
5. leopard
6. monkey\_prosimian
7. rodent



Note that each image should be associated with one class, since each image contains at most one animal. (Taï National Park is of course a great place to make friends, but we have chosen only those images that capture a single species class). The train\_labels.csv file therefore has a single row per image, containing the following values:

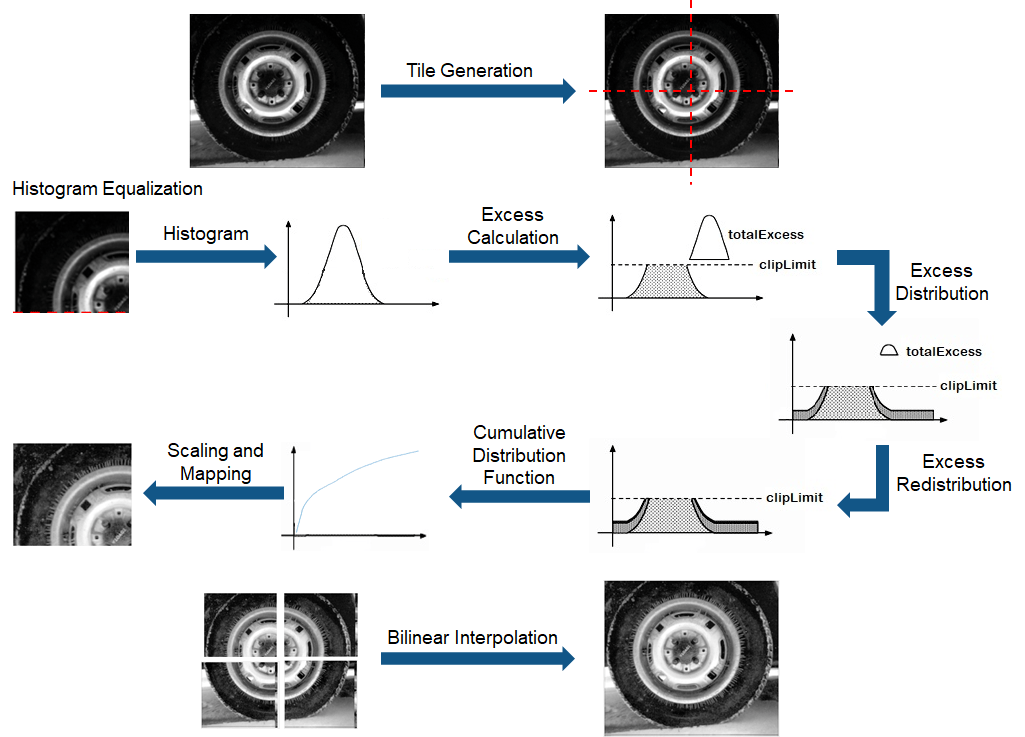
[Image Deblurring](https://services.saiwa.ai/base-service/deblurring) refers to recovering a sharp image from the input blurred one. It is inherently an ill-posed problem that has been studied in many researches during the past few decades. The blurring of an image can be caused by many factors, like: camera or subject movement during the image acquisition, out-of-focus optics, or scattered light distortion. Numerous image deblurring methods have been provided from classic algorithms with mathematical principle to recent methods based on machine learning and [deep learning](https://services.saiwa.ai/learning/deep-learning/job) advances.

What is deblurring in image processing?

Deblurring is **the process of removing blurring artifacts from images**. Deblurring recovers a sharp image S from a blurred image B, where S is convolved with K (the blur kernel) to generate B. Mathematically, this can be represented as. (where \* represents convolution).

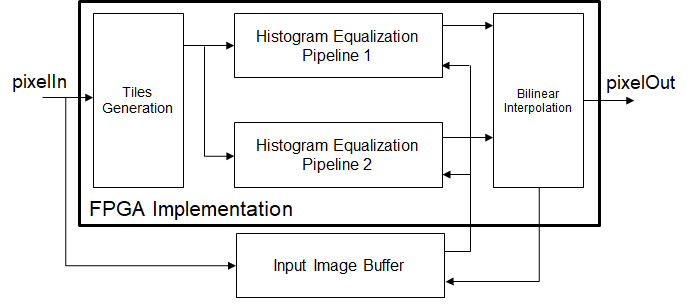
Adaptive histogram equalization (AHE) is an image pre-processing technique used to improve contrast in images. It computes several histograms, each corresponding to a distinct section of the image, and uses them to redistribute the luminance values of the image. It is therefore suitable for improving the local contrast and enhancing the definitions of edges in each region of an image. However, AHE has a tendency to overamplify noise in relatively homogeneous regions of an image. A variant of adaptive histogram equalization called contrast-limited adaptive histogram equalization (CLAHE) prevents this effect by limiting the amplification.

### CLAHE Algorithm



The CLAHE algorithm has three major parts: tile generation, histogram equalization, and bilinear interpolation. The input image is first divided into sections. Each section is called a tile. The input image shown in the figure is divided into four tiles. Histogram equalization is then performed on each tile using a pre-defined clip limit. Histogram equalization consists of five steps: histogram computation, excess calculation, excess distribution, excess redistribution, and scaling and mapping using a cumulative distribution function (CDF). The histogram is computed as a set of bins for each tile. Histogram bin values higher than the clip limit are accumulated and distributed into other bins. CDF is then calculated for the histogram values. CDF values of each tile are scaled and mapped using the input image pixel values. The resulting tiles are stitched together using bilinear interpolation, to generate an output image with improved contrast.

### HDL Implementation



This figure shows the block diagram of the HDL implementation of the CLAHE algorithm. It consists of a tile generation block, a histogram equalization pipeline block, a bilinear interpolation block, and an input image buffer block. Tiles are generated by modifying the pixelcontrol bus of the pixel stream for the desired tile size. The pixel stream and the modified pixelcontrol bus are fed to the histogram equalization pipeline. Two histogram equalization pipelines are required to keep pace with the input data. They operate in ping-pong manner. Each pipeline contains histogram equalization modules equal to the number of tiles in the horizontal direction. The histogram equalization modules work in parallel to compute histogram equalization for each tile. The last stage in the histogram equalization module, scaling and mapping, needs the original input image data. This data is stored in an input image buffer block. The bilinear interpolation block generates addresses to read the input image pixel values from the memory. The input image pixel values from the image buffer block are given to the histogram equalization modules for mapping. Mapped values obtained from histogram equalization are scaled and used in the bilinear interpolation computation to reduce boundary artifacts.