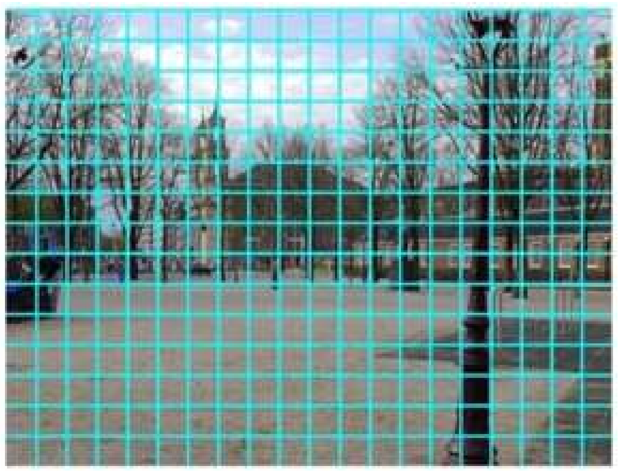
Preprocess Image Data For Machine Learning

AnisKHELOUFI Follow
Mar 28 · 5 min read ★

In recent years, there have been huge advances in the field of neural networks specifically to train models for image recognition and classification. To design and implement an efficient and powerful image classification solution. You need to know how such image data can be read into your program, and how you can represent images as tensors to feed into your model to get the best out of ML algorithms.

Image representation:

All images can be represented using pixels. images are essentially just matrices where the individual cells of the matrix contain pixel data. An image can be divided into a grid of pixels. Each element of this grid is used to hold pixel information, and the value of a pixel depends on the type of image that you're working with,This pixel representation of image matrices can be fed into a machine learning model such as those built using neural networks. These neural networks will identify edges, colors, and shapes within the images, and then perform image recognition. It can perform classification or identify and detect objects within an image.



Color image:

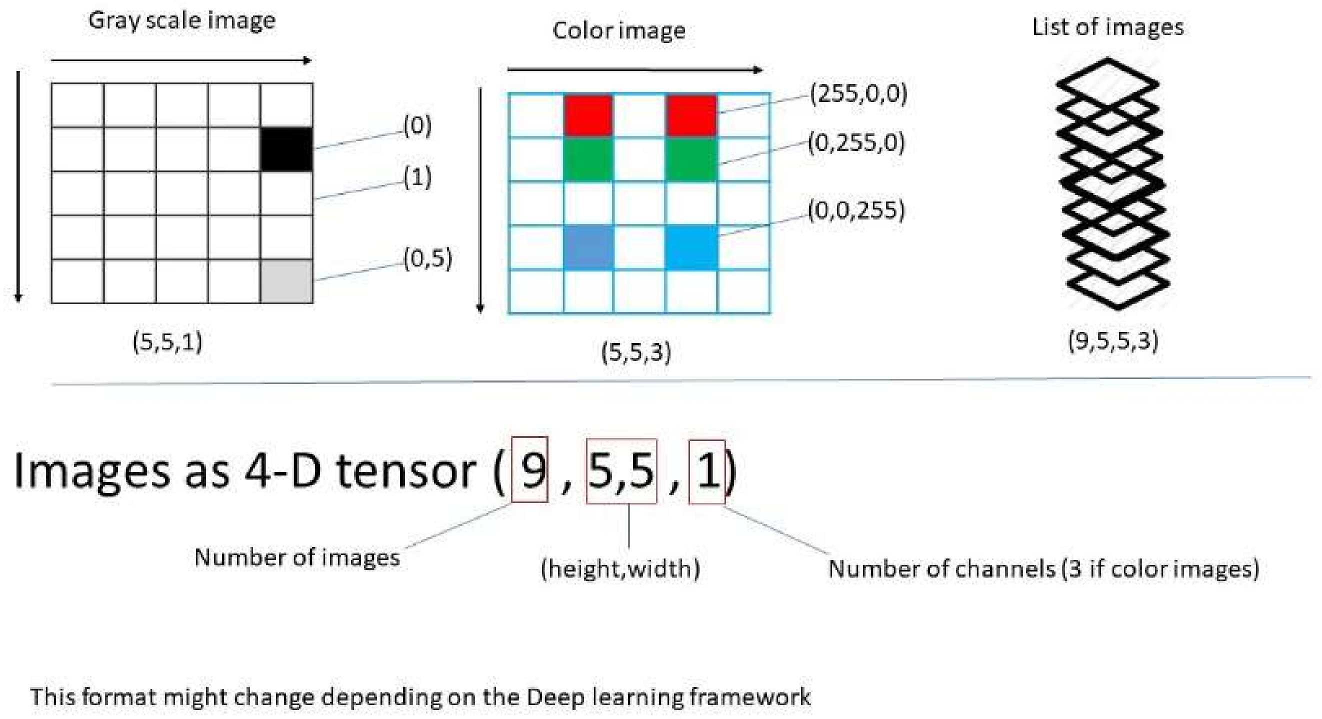
Color or RGB image, every pixel is represented using R, G, and B values. This means that you need three numbers to represent an image, a pixel that is red in color will be represented as 255, 0, 0. The R value is 255, G and B are 0's this is a 3-channel image or a multi-channel image. you'll find that neural networks work better when the numeric values are small, so these pixel values in the range 0–255 are often scaled to be in the 0–1 range

Gray scale images:

grayscale images, Single channel ;one value to represent a pixel, and that is the intensity of a pixel. Each pixel represents only intensity information, so you have one value in the range 0–1. A value of 1 means a pixel with the highest intensity, 0 represents a pixel with no intensity at all.

Image as a Tensor:

images are essentially just 3-dimensional tensors. Images can be represented using a 3D matrix. The number of channels that you have in an image specifies the number of elements in the third dimension. The first two dimensions, refer to height and width of an image. When you're working with neural networks and images, you'll feed in a batch or a list of images to your neural network at a time. A batch of images can be represented as one 4-D tensor, where the first dimension is the batch dimension. This, of course, means that the all of the images in that batch should be of the same size; same height, same width, and same number of channels.


Gray scale image
(0)
(1)
(0,5)
(5,5,1)
Color image
(255,0,0)
(0,255,0)
(0,0,255)
(5,5,3)
List of images
(9,5,5,3)
Images as 4-D tensor (9 , 5,5 , 1)
Number of images
(height,width)
Number of channels (3 if color images)
This format might change depending on the Deep learning framework

Most common Pre-Processing techniques :

There are a number of pre-processing techniques that you can apply in order to make your models more robust. These are the most common used to improve the performance of your CNN.

- **Aspect ratio:**

The aspect ratio of an image is the ratio of the width of your image to the height of your image, you should scale all of your images so that they have a uniform aspect ratio. most models assume that the shape of your input image is square, you should crop your images so that they are all in the square shape, you might want to perform a center crop to extract the most important part of your image. Cropping an image so that we get a square shape of the most important features, makes the aspect ratio of all images that you are working with constant.

Resizing and reshaping of an RGB image

Cropping qnd denoising

- **Unifrom image size :**

Your image data is fed in N batches to train your model. A batch of images requires that all images in your data be of the same size. You should ensure that all images fed into your model have the same height and width. Convolutional neural networks use something called feature maps to extract information from the input image. You may need to rescale your images to fit your image size to the feature maps that you plan to use. PyTorch, offers a number of utility functions that you can use to perform these rescaling operations.

- **Mean and perturbed images :**

A mean image is basically when you calculate the average pixel value across the entire training data set. The use of a perturbed image is where you intentionally distort pixels by varying them from the mean image. This can make your convolutional neural network more robust by preventing it from focusing only on the center of your image. It'll force your CNN to look at other parts of the image.

- **Normalized image input:**

Your neural networks will train faster if you work with normalized input images. Normalizing your input image means that you center your pixel values around 0 mean. You normalize each pixel by subtracting the mean pixel value across all of your training data or a single batch, and dividing by the standard deviation. This ensures that each pixel has a similar data distribution. After normalization, you might choose to rescale your pixel values to be in the 0–1 range.

- **Dimentionality reduction** :

Images that you work with can be very large, which means you are dealing with features of very high dimensionality. You might want to perform dimensionality reduction before you use images to train your model. Reducing RGB images to grayscale is a kind of dimensionality reduction. dimensionality reduction helps extract the latent of significant features from your underlying data so your neural network does not have to deal with irrelevant features.

- **Data augumentation :**

perturbed images are a form of data augmentation, where you intentionally distort pixel values from the mean. Other data augmentation techniques are where you scale, rotate or perform affine transforms on the input image. Augmentation techniques applied to the input makes your convolutional neural network more robust, and it reduces the risk of overfitting your convolutional neural network on your training data.