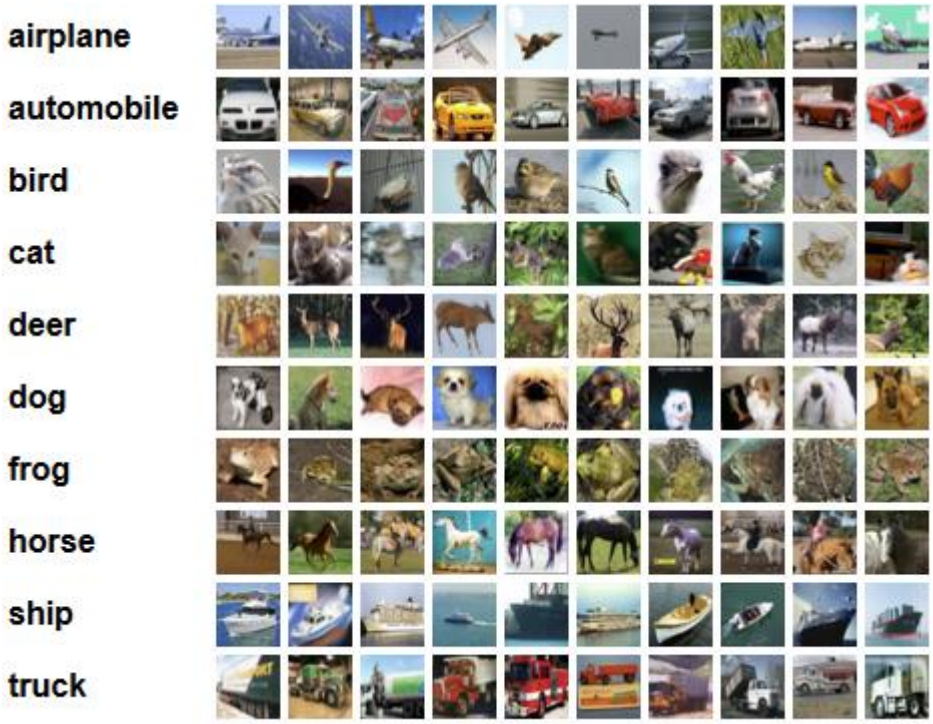
https://www.cs.toronto.edu/~kriz/cifar.html

The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

Download

If you're going to use this dataset, please cite the tech report at the bottom of this page.

Version	Size	md5sum

CIFAR-10 python version 163 MB c58f30108f718f92721af3b95e74349a

Dataset layout

Python / Matlab versions

I will describe the layout of the Python version of the dataset. The layout of the Matlab version is identical.

The archive contains the files data_batch_1, data_batch_2, ..., data_batch_5, as well as test_batch. Each of these files is a Python "pickled" object produced with cPickle. Here is a python2 routine which will open such a file and return a dictionary:

```
def unpickle(file):
    import cPickle
    with open(file, 'rb') as fo:
        dict = cPickle.load(fo)
    return dict
```

And a python3 version:

```
def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict
```

Loaded in this way, each of the batch files contains a dictionary with the following elements:

- **data** -- a 10000x3072 numpy array of uint8s. Each row of the array stores a 32x32 colour image. The first 1024 entries contain the red channel values, the next 1024 the green, and the final 1024 the blue. The image is stored in row-major order, so that the first 32 entries of the array are the red channel values of the first row of the image.
- **labels** -- a list of 10000 numbers in the range 0-9. The number at index *i* indicates the label of the *i*th image in the array **data**.

The dataset contains another file, called batches.meta. It too contains a Python dictionary object. It has the following entries:

- **label_names** -- a 10-element list which gives meaningful names to the numeric labels in the **labels** array described above. For example, label_names[0] == "airplane", label_names[1] == "automobile", etc.

Binary version

The binary version contains the files data_batch_1.bin, data_batch_2.bin, ..., data_batch_5.bin, as well as test_batch.bin. Each of these files is formatted as follows:

<1 x label><3072 x pixel>

...

<1 x label><3072 x pixel>

In other words, the first byte is the label of the first image, which is a number in the range 0-9. The next 3072 bytes are the values of the pixels of the image. The first 1024 bytes are the red channel values, the next 1024 the green, and the final 1024 the blue. The values are stored in row-major order, so the first 32 bytes are the red channel values of the first row of the image.

Each file contains 10000 such 3073-byte "rows" of images, although there is **nothing delimiting the rows**. Therefore each file should be exactly 30730000 bytes long.

There is another file, called batches.meta.txt. This is an ASCII file that maps numeric labels in the range 0-9 to meaningful class names. It is merely a list of the 10 class names, one per row. The class name on row *i* corresponds to numeric label *i*.