Initialization

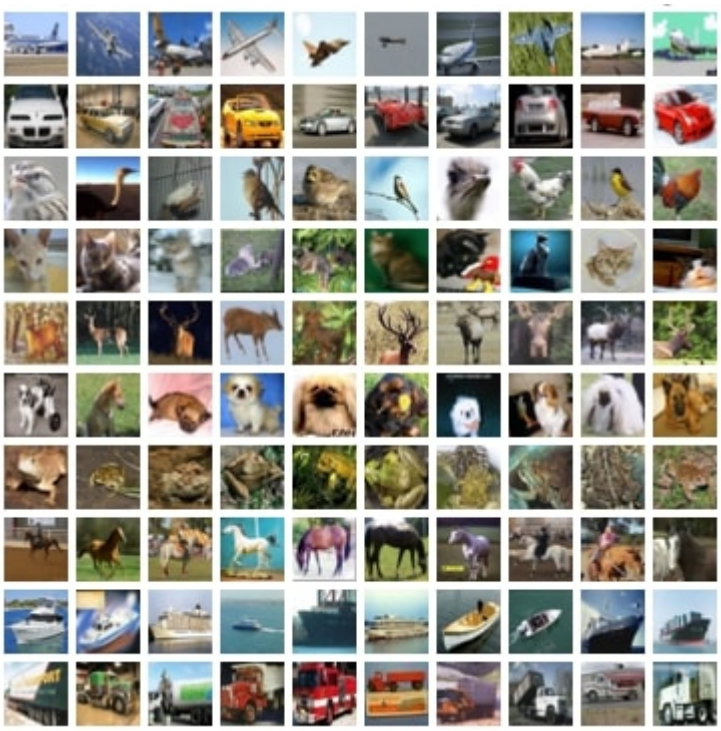
Learn about the CIFAR-10 dataset along with the input and output sizes of the data.

Chapter Goals:

- Learn about the CIFAR-10 dataset
- Initialize the model with data dimensions

A. CIFAR-10

The CIFAR-10 (Canadian Institute for Advanced Research) dataset contains 60,000 color images with dimensions 32x32. The images are distributed evenly across 10 categories: **airplane**, **automobile**, **bird**, **cat**, **deer**, **dog**, **frog**, **horse**, **ship**, **truck**. We split the dataset into 50,000 images for training and 10,000 images for testing.



Example images from the CIFAR-10 dataset.

The CIFAR-10 dataset is available for download (along with the 100 category version, CIFAR-100), on Alex Krizhevsky's website.

(Fun fact: Alex Krizhevsky helped invent the AlexNet model mentioned in the previous chapter, which incidentally was named after him. The LeNet model from the **CNN** section was also named

after its inventor, Yann LeCun. SqueezeNet, however, is not named after its developers.)

B. Inputs and labels

Each image has dimensions 32x32, meaning `original_dim` will be 32 for the CIFAR-10 dataset. The images have color, so they follow the RGB format, meaning each pixel contains three integers (one each for the red, blue, and green channels). In total, an image is represented by

$$32 \times 32 \times 3 = 1024 \times 3 = 3072$$

integers between 0 and 255. The first 1024 integers represent the red channel pixel values, the next 1024 represent the blue channel pixel values, and the final 1024 represent the green channel pixel values.

The labels this time are just single integers corresponding to the class index of the image, rather than one-hot vectors. This is referred to as a *sparse representation* of the labels. However, `output_size` is still equal to the number of image categories, in this case 10.

For a batch of input data, the shape of `inputs` is `(batch_size, 3 * self.original_dim**2)` and the shape of `labels` is `(batch_size,)`, where `batch_size` represents the size of the batch. Due to the sparse representation, `labels` is a 1-D tensor (1-D tensors have a trailing comma in their shape).

```
batch_size = 32
dataset = dataset.batch(batch_size)
it = dataset.make_one_shot_iterator()
inputs, labels = it.get_next()
with tf.Session() as sess:
    # Batch of data size 10
    input_arr, label_arr = sess.run(
        (inputs, labels))
```

Example batch of input data and labels. Note that dataset contains all the CIFAR-10 data and corresponding labels.

In the example, `inputs` represents the input data tensor, while `labels` represents the 1-D label tensor. The batch size is set to 32 in the example.

Time to Code!

In this section of the course you'll be creating a Python class, `SqueezeNetModel` that represents the SqueezeNet model you'll be building.

We'll set the original height/width dimension of the image data (`original_dim`) and the number of classes (`output_size`).

Inside the `__init__` function, set `self.original_dim` equal to `original_dim` and set `self.output_size` equal to `output_size`.

In the next chapter we'll be doing image processing, so we need to set the resized dimension of the images.

Set `self.resize_dim` equal to `resize_dim`.

```
import tensorflow as tf

class SqueezeNetModel(object):
    # Model Initialization
    def __init__(self, original_dim, resize_dim, output_size):
        # CODE HERE
        pass
```