practical advice for images

- tensorflow comes with a bunch of good pretrained models (eg. for image recognition). it also has a bunch of experimental research models.
- An advantage of transfer learning, other than good regularizing and guidance for optimization is that it can be run in as little as thirty minutes on a laptop, without requiring a GPU.
- 'Bottleneck' is an informal term we often use for the layer just before the final output layer that actually does the classification. while doing transfer learning style last layer retraining, we can cache the outputs of the layers upto (and excluding) the first layer that will be retrained.
- things to keep in mind while collecting the dataset:
  - gather as much data as you can about each category you want to be recognized
  - gather training data that is a good representation of what the application will see
  - gather a wide variety of training data. avoid the tank recognition problem.
  - It might be worth splitting big categories that cover a lot of different physical forms into smaller ones that are more visually distinct. For example instead of 'vehicle' you might use 'car', 'motorbike', and 'truck'.
- CIFAR-10 classification is a common benchmark problem in machine learning. The problem is to classify RGB 32x32 pixel images across 10 categories:

```
airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.
```

- The TensorFlow `layers` module provides a high-level API that makes it easy to construct a neural network. It provides methods that facilitate the creation of dense (fully connected) layers and convolutional layers, adding activation functions, and applying dropout regularization.
- We can use `tf.layers` to construct a model spec and return a `tf.estimator.EstimatorSpec(mode=mode, loss=loss,eval_metric_ops=eval_metric_ops)` This can then be used to construct a `tf.estimator.Estimator`, which can be trained and evaluated with higher level estimator APIs.
- multi GPU architecture:
  - Naively employing asynchronous updates of model parameters leads to sub-optimal training performance because an individual model replica might be trained on a stale copy of the model parameters. Conversely, employing fully synchronous updates will be as slow as the slowest model replica.
  - In a workstation with multiple GPU cards, each GPU will have similar speed and contain enough memory to run an entire CIFAR-10 model. Thus, we opt to design our training system in the following manner:
    - Place an individual model replica on each GPU.
    - Update model parameters synchronously by waiting for all GPUs to finish processing a batch of data.