Dataset Loader (PyTorch & TensorFlow Comparison)

Dataset loading is the first and most fundamental step in a deep learning training pipeline.
Both PyTorch and TensorFlow (Keras) provide built-in APIs for loading image datasets, but their usage patterns are different.

This document compares dataset loading approaches in PyTorch and TensorFlow while assuming the same directory structure.

1. Directory Structure Assumption

Both frameworks assume the following directory structure:

```
dataset/train/
            class_1/
                    img1.jpg
                    img2.jpg
            class_2/
                    img1.jpg
                    img2.jpg
```

Class labels are inferred automatically from subdirectory names.

2. PyTorch Dataset Loader

PyTorch typically uses torchvision.datasets.ImageFolder together with DataLoader.
ImageFolder reads images from a directory structure where each subdirectory represents a class.
DataLoader handles batching, shuffling, and multi-threaded loading.

Example usage:

```
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
    transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])

train_dataset = datasets.ImageFolder(
    "dataset/train",
    transform=transform
)

train_loader = DataLoader(
    train_dataset,
    batch_size=32,
    shuffle=True,
    num_workers=4,
    pin_memory=True
)
```

3. TensorFlow (Keras) Dataset Loader

TensorFlow (Keras) provides the image_dataset_from_directory API, which simplifies dataset loading into a single function call.
This API assumes the same directory structure as PyTorch ImageFolder.

Example usage:

```
import tensorflow as tf
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "dataset/train",
    image_size=(224, 224),
    batch_size=32,
    shuffle=True
)
```