

Das Modul **torchvision.datasets** ist die eingebaute Schatzkiste von PyTorch für Bilddaten. Es enthält eine riesige Auswahl an **vorgefertigten, populären Datensätzen** für das maschinelle Sehen, die du mit nur einer Zeile Code herunterladen, formatieren und direkt in dein neuronale Netzwerk einspeisen kannst.

Die 3 großen Vorteile

- 📦 **Kein manuelles Herunterladen:** Kein Suchen auf Webseiten, kein Entpacken von ZIP-Dateien. Python erledigt das im Hintergrund.
- ⚙️ **Direkte Vorverarbeitung:** Du kannst transforms (wie Skalierung oder Normalisierung) direkt beim Laden ankoppeln.
- 🌱 **Einheitliches Interface:** Egal ob MNIST, CIFAR oder ImageNet – alle Datensätze verhalten sich im Code exakt gleich.

Die bekanntesten Datensätze im Überblick

Die Bibliothek deckt verschiedene Kategorien von Computer-Vision-Aufgaben ab:

Kategorie	Datensätze (Beispiele)	Typische Nutzung
Klassifizierung (Einfach)	MNIST, FashionMNIST, CIFAR10, CIFAR100	Einstieg, Testen von Modellen, Benchmarks
Klassifizierung (Groß)	ImageNet, Caltech101, CelebA	Echtes Deep Learning, Erkennung komplexer Objekte
Objekterkennung / Segmentierung	CocoDetection, VOCSegmentation	Finden von Boxen (Bounding Boxes) oder Pixelmasken
Eigene Bilder	ImageFolder	Der wichtigste Datensatz für deine eigenen, lokalen Fotos

Praxis-Beispiel: Datensatz laden und transformieren

Hier siehst du, wie du den bekannten **CIFAR-10** Datensatz (60.000 kleine Farbbilder in 10 Klassen) lädst und für dein Modell vorbereitet:

```

from torchvision import datasets, transforms

# 1. Pipeline für die Bildverarbeitung definieren
transformations = transforms.Compose([
    transforms.ToTensor(),          # Bild in PyTorch-Tensor
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)) #
    # Helligkeit anpassen
])

# 2. Trainingsdaten herunterladen und laden
train_dataset = datasets.CIFAR10(
    root='./data',                # Ordner, in dem die Daten gespeichert
    train=True,                   # Lädt den Trainings-Split (False lädt
    download=True,                # Lädt es aus dem Internet herunter,
    transform=transformations     # Wendet unsere Pipeline oben an
)

# 3. Testen, wie viele Bilder da sind
print(f"Anzahl der Trainingsbilder: {len(train_dataset)}") #
# Ausgabe: 50000

```

Der Lebensretter für eigene Projekte: datasets.ImageFolder

Wenn du **deine eigenen Bilder** von der Festplatte trainieren willst, musst du keinen eigenen Code schreiben. Du musst deine Fotos nur in Ordnern sortieren, die so heißen wie deine Klassen:

```

mein_datensatz/
├── katzen/
│   ├── katze1.jpg
│   └── katze2.jpg
└── hunde/
    ├── hund1.jpg
    └── hund2.jpg

```

Diesen Ordner liest PyTorch mit einer einzigen Zeile ein und weist die Labels automatisch zu:

```
mein_dataset = datasets.ImageFolder(root='mein_datensatz/',  
transform=transformations)  
print(mein_dataset.classes)  # Ausgabe: ['hunde', 'katzen']
```

Möchtest du einen der **Standard-Datensätze** für ein bestimmtes Experiment nutzen, oder planst du, deine **eigenen Bilder über ImageFolder** einzulesen? Ich kann dir für beides das passende Code-Gerüst zeigen!