Video Classification - Data layer

Datasets — Benchmarks & Sources

Kinetics-400 / 600 / 700

- **What it is:** Large-scale human action datasets (240–650k YouTube clips, ~10s each) with 400/600/700 classes.
- **Why it matters:** The standard for pretraining and benchmarking video models; broad, diverse actions.
- **Quirks:** YouTube links occasionally dead; clip quality varies; class granularity uneven.
- **Where:** Hugging Face (`kinetics700`, community mirrors), Google AI original (YouTube links).

Something-Something V2 (20BN)

- **What it is:** ~220k crowd-sourced clips emphasizing object–motion interactions (e.g., "pushing something from left to right").
- **Why it matters:** Tests temporal reasoning over appearance (strong for transformer baselines).
- **Quirks:** Classes are fine-grained templates; appearance cues help less than motion cues.
- **Where:** Hugging Face (`something-something-v2` mirrors), 20BN website.

UCF101

- **What it is:** 13k videos across 101 action classes (sports, playing instruments, etc.).
- **Why it matters:** Lightweight, classic benchmark; fast experiments and ablations.
- **Quirks:** Small, somewhat biased; many models now saturate; useful for transfer learning demos.
- **Where:** Hugging Face (`ucf101`).

HMDB51

- **What it is:** ~7k videos, 51 classes from movies/YouTube with realistic motion.
- **Why it matters:** Complement to UCF101; harder due to noise and variability.
- **Quirks:** Small size; label noise; performance highly sensitive to sampling.
- **Where:** Hugging Face (`hmdb51`).

Moments in Time

- **What it is:** ~1M 3-second clips, 339 classes capturing brief "moments".
- **Why it matters:** Scale + short duration stresses temporal discrimination in tight windows.
- **Quirks:** Very short clips; ambiguous context; heavy class imbalance in the wild.
- **Where:** Project site (community HF mirrors exist).

ActivityNet

- **What it is:** ~20k videos, 200 classes, with temporal activity annotations (trimmed & untrimmed).
- **Why it matters:** Bridges classification and temporal localization; longer web videos.
- **Quirks:** Long, untrimmed videos require proposal/sampling strategies.
- **Where:** Project site (HF mirrors like `activitynet`).

EPIC-KITCHENS (2018/2020/2022)

- **What it is:** Large egocentric (first-person) dataset of everyday kitchen activities (verbs/nouns/actions).
- **Why it matters:** Tests fine-grained, egocentric understanding and verb–noun compositionality.
- **Quirks:** Long videos, multiple labels (verb/noun/action), domain very specific.
- **Where:** Project site; some HF subsets.

Charades

- **What it is:** ~10k indoor videos with multi-label annotations for household activities.
- **Why it matters:** Multi-label classification + localization; realistic indoor scenes.
- **Quirks:** Co-occurring actions; label sparsity; class imbalance.
- **Where:** Project site; community HF mirrors.

HACS (Human Action Clips & Segments)

- **What it is:** ~1.5M clips plus 139k segments for 200 actions (YouTube), curated for action understanding.
- **Why it matters:** Scale for pretraining; supports classification + localization.
- **Quirks:** YouTube availability; label noise despite curation.
- **Where:** Project site (community mirrors).

AVA (for context)

- **What it is:** Atomic visual actions with spatiotemporal person boxes at 1 Hz.
- **Why it matters:** More for detection/localization, but often used to pretrain features for classification.
- **Quirks:** Dense annotations, heavier pipelines.
- **Where:** Project site.

Kinetics-Mini / SSv2-Mini (community)

- **What it is:** Small curated subsets (few classes) of Kinetics/SSv2.
- **Why it matters:** Rapid prototyping on laptops; sanity checks before scaling.
- **Quirks:** Not official; distribution shift vs full sets.
- **Where:** Hugging Face (search “mini kinetics”, “mini ssv2”).

YouTube-8M (labels only)

- **What it is:** Millions of YouTube IDs with precomputed features & labels (multi-label).
- **Why it matters:** Large-scale weakly supervised learning and multi-label tagging.
- **Quirks:** Access to original videos is unreliable; use precomputed features.
- **Where:** Google AI site; TFRecords/features widely mirrored.

> **Note on OpenAI:** OpenAI does not distribute public video datasets; use HF/Google/academic sources above.

Preprocessing (what to do and why)

Decoding & Frame Sampling

We turn compressed video into a fixed-length clip of frames suitable for the model.

- **Uniform / strided sampling:** Pick `T` frames evenly across the clip → Stabilizes coverage of the action; reduces bias toward any segment.
- **Random temporal jitter:** Randomize start index & stride per epoch → Improves temporal robustness; acts like augmentation in time.
- **FPS normalization:** Decode at a target fps (e.g., 30→16/24 fps) → Keeps motion dynamics consistent across videos/devices.
- **Clip length (`T`):** Common: 8, 16, 32 frames (sometimes 64) → Balance between temporal context and compute.

Spatial Resizing & Cropping

We make frames a consistent size while adding stochasticity for generalization.

- **Train:** `RandomResizedCrop(224)` + `RandomHorizontalFlip()` → Standard ImageNet-like pipeline adapted per-frame for video.
- **Eval:** `Resize(256)` + `CenterCrop(224)` → Deterministic evaluation to fairly compare runs.

Color & Photometric Augmentations

We diversify appearance to prevent overfitting to lighting/camera artifacts.

- **ColorJitter / RandAugment (light):** Small brightness/contrast/saturation/hue → Videos are sensitive; keep mild to avoid temporal flicker.
- **GaussianBlur (light), RandomGrayscale (rare):** → Only slight usage; strong photometric shifts can harm motion cues.

Temporal Augmentations

We alter timing to improve motion invariance without breaking semantics.

- **Temporal crop/jitter:** Sample contiguous `T` with random start → Encourages learning from different subevents.
- **Speed jitter (e.g., 0.9–1.1×):** Slight playback rate changes → Teaches rate invariance; avoid large factors to keep labels valid.

Normalization

We adjust pixel values so they're centered and scaled, making training stable.

- **ImageNet stats:** `Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])` → Matches the preprocessing expected by most pretrained models.
- **From scratch:** `Standardize per dataset` → If no pretrained weights, compute dataset mean/std to reduce covariate shift.

Tokenization / Feature Extraction (video-specific)

We convert frames/clips into the input tokens/features expected by the model.

- **Patch tokenization (ViT/TimeSformer/ViViT):** Split each frame into patches (e.g., 16×16), then add temporal/positional embeddings → Enables spatiotemporal self-attention over patch tokens.
- **Tubelet embedding (VideoMAE/ViViT):** 3D patches across space–time → Models motion with fewer tokens than per-frame patches.
- **Optical flow / motion vectors (optional):** Precompute flow (e.g., TV-L1) → Helps when motion is crucial, but increases pipeline cost.
- **Audio features (AV tasks):** Extract log-mel spectrograms / MFCCs aligned to frames → Boosts performance for actions with characteristic sounds.

Dataset-Specific Quirks

- **Kinetics:** Link rot; use mirrors & caching; variable resolutions/aspect ratios.
- **SSv2:** Motion > appearance; avoid overly strong color aug; prioritize temporal jitter; longer `T` helps.
- **UCF101/HMDB51:** Small → heavy regularization & pretraining recommended; strong augment + mixup/cutmix can help.
- **ActivityNet/Charades:** Untrimmed & multi-label—decide if you train on trimmed clips or use segment proposals; evaluation often mAP not top-1.
- **EPIC-KITCHENS:** Verb–noun labels; consider multi-head outputs; egocentric perspective favors wider FOV crops and higher `T`.

Dataloading tips

We prepare the dataset so training is fast, reproducible, and efficient.

- **Use efficient decoders:** Prefer **Decord**, **PyAV**, or `torchvision.io` over naive OpenCV → Faster, safer multi-worker decoding; fewer deadlocks.

- **Prefetch & pin memory:** `DataLoader(pin_memory=True, prefetch_factor>1)` → Ensures GPUs aren't starved while waiting for data.
- **Batching layout:** Shape `(B, T, C, H, W)` for Transformers; `(B, C, T, H, W)` for some CNNs → Match your model's expected tensor order to avoid silent bugs.
- **Worker init functions:** `worker_init_fn=seed_all` → Makes sure random augmentations are reproducible across runs.
- **Deterministic validation:** `Resize(256) + CenterCrop(224) + Normalize(...)` → Keeps evaluation transforms fixed for fair comparisons.
- **Cache decoded frames (optional):** On SSD as npy/pt tensors for small datasets → Big speedup for UCF101/HMDB51 when iterating quickly.
