Episode Book

EPISODE #6706

How Accidental Dataset Formats Became AI Standards

PUBLISHED	20 November 2026 at 13:37 UTC
DURATION	28:32
LISTEN	myweirdprompts.com/episode/accidental-ai-dataset-formats-44944/
HOSTS	Corn, Herman, Dorothy, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.36
SOURCES	25 cited by the research (listed at the end)
BOOK GENERATED	20 November 2026 at 21:20 UTC

SYNOPSIS

No standards body designed the dataset formats powering modern AI. They survived because a library shipped a loader or a benchmark needed to score fairly. This breakdown compares speech and image formats across file organization, geometry, and extra flags—from LJSpeech's pipe-delimited CSV and NVIDIA's NeMo JSONL manifests to Mozilla Common Voice, DeepSpeech, COCO, Pascal VOC, and YOLO. Along the way, discover why COCO's category IDs still have eleven load-bearing holes and how trainer batching shaped speech schemas.

TRANSCRIPT



Corn

Thirteen thousand one hundred clips. One speaker. Seven non-fiction books published between eighteen eighty-four and nineteen sixty-four, read aloud by a LibriVox volunteer in twenty sixteen and seventeen, and somehow that one person's reading voice became the folder layout half the TTS world copies.



Herman

And Daniel, as it happens, has been swimming in exactly this water. He's been doing his ASR fine-tuning, he's hit the point where "there are conventions" stops being useful advice and starts being a question, and he's written in asking about the named end of the spectrum. The specific formats with actual folder layouts and manifests and annotation schemas. He half-remembers the speech ones, thinks maybe LJSpeech or the NeMo JSONL manifest, and he wants the images too, COCO, YOLO, Pascal VOC, plus whatever the 3D datasets on Hugging Face are converging on.



Corn

And the honest answer to his half-memory is that he remembered right. But before we get into folder trees, we should be clear about what a format actually is, because the line he's drawn is the whole episode.



Herman

The line is advice versus contract. A convention says put your wavs in a folder, name things sensibly, keep a list somewhere. A named format says your metadata file is pipe-delimited with exactly three columns, the third column spells numbers out as words, and your audio is mono sixteen-bit PCM at twenty-two thousand fifty hertz, and your training code will enforce all of it or refuse to run. One is a recommendation from a friend. The other is a contract with a machine.



Corn

Pipe-delimited. In a file called metadata dot csv. That's the first sign nobody designed this.



Herman

Nobody designed any of them! That's the thesis before we even start. I was trying to think of a single format we're about to discuss that was created by a standards body, and I can't find one. Every one of these is a fossil of a tool or a benchmark that needed it at the time. The survivors survived because a library shipped a loader, not because a committee blessed them.



Corn

So how do we compare them fairly, if they're all accidents?



Herman

Three axes, and this is a useful frame. Every annotation format fixes three things. First, file organization — one big file for the whole dataset, one file per image, or one file per object. Second, how geometry is written — corners versus centres, pixels versus fractions, zero-based versus one-based. Third, which extra flags exist — crowd, difficult, truncated, void. Take any two formats and they differ on all three axes at once.



Corn

Speech formats don't have geometry, though. What replaces those axes?



Herman

For speech it's more like — where does the transcript live, what does the trainer need to know about each clip before it can batch it, and how much schema is there to break. Speech formats are shaped by the trainer that consumes them. Image formats are shaped by the benchmark that scores them. That difference turns out to explain almost everything.



Corn

Then let's start with the trainer side. The clearest example of a format becoming a default almost by accident is the one that became the template for every TTS project since.



Herman

LJSpeech. And the tree is almost embarrassingly simple. The root folder has a README, a metadata file, and a folder called wavs containing the clips. That's it.



Corn

And the metadata file is the format. Three fields.



Herman

Three fields, pipe-delimited: the clip ID, the raw transcription, and the normalized transcription. The ID is just the wav filename without its extension, so LJ zero zero one dash zero zero zero one points at LJ zero zero one dash zero zero zero one dot wav. Column two is what the speaker actually read. Column three is the same text with every number, ordinal and monetary unit expanded into full words, so the model never has to guess how "one hundred and twelve" sounds.



Corn

Which is the part people assume is a cleanup pass.



Herman

It's the part people assume is a cleaned-up copy of column two, and it isn't — it's a normalization field, and for TTS it's arguably the more important of the two, because the model trains on the spoken form. Get column three wrong and you've taught the model that "1964" is pronounced as digits.



Corn

I'd like to lodge a formal complaint about a format whose file extension lies about its delimiter. The comma in csv stands for comma.



Herman

The usual story is that transcriptions contain commas, so they picked a pipe. The deeper story is that this whole layout was one person, Keith Ito, building a clean single-speaker corpus, and torchaudio then shipped a loader for it with a pinned checksum, and that was the whole adoption process. There was no campaign. A library said "we can load this" and the format became the bring-your-own-data template. The State Library of Queensland did a survey of options in twenty twenty-three and settled on LJSpeech's layout precisely because it's what the speech synthesis libraries expect.



Corn

Because the alternative is writing your own dataset class.



Herman

Nobody writes their own dataset class if they can possibly avoid it, and that's not laziness, that's the actual selection pressure. The format that wins is the one where the loader already exists. We did an episode a while back on the safetensors side of this — weight formats rather than data formats — and the same rule held there: the format that ships in the default toolchain wins regardless of its technical elegance.



Corn

LJSpeech's three fixed columns are perfect for single-speaker TTS and useless for anything else. Which is the gap the manifest format was built to fill.



Herman

NeMo, from NVIDIA. One JSON object per line, in a plain text file with one manifest per split. You pass it to the trainer as a manifest file path. The minimal ASR record is three fields: audio filepath, text, duration. That last one, duration in seconds, is there so the loader can bucket and batch without probing every file first.



Corn

And the rules that bite people.



Herman

Two, and they both bite quietly. No blank lines anywhere — a stray empty line at the end of the file and things go wrong in ways that depend on the reader. And one manifest per split, full stop. The docs are unusually blunt about why: if you point training at more than one manifest, you will be loading validation data with your training data and vice versa, and your metrics become fiction.



Corn

Which is a bug you only find when the model is suspiciously good.



Herman

Suspiciously excellent validation scores, yes. That's the one. And what makes NeMo's format different from LJSpeech is that the schema is versioned by model family. The minimal record is the floor, but Canary version one requires source language, target language, task and punctuation — and Canary Flash and version two drop task entirely and make punctuation, inverse text normalization, timestamps, diarization, decoder context and emotion all optional. Audio-to-audio tasks swap in input filepath and target filepath instead. The format anticipates that the model on top of it will evolve.



Corn

A JSON line absorbs a new key and shrugs. A pipe-delimited csv with three columns has nowhere to put it.



Herman

That's the entire architectural difference. And it's not theoretical — NVIDIA's Granary dataset on Hugging Face uses the manifest convention at scale, across languages and corpora, with extra fields like taskname, utterance ID and dataset source bolted on, and nothing breaks, because a JSON line doesn't care. LJSpeech's three fixed columns cannot do any of that, and they don't need to. That's why NeMo's format owns ASR and LJSpeech's owns TTS.



Corn

Two more speech formats that earn a mention, and I suspect both are one-project formats.



Herman

They are, and that's the interesting thing about them. Mozilla Common Voice is a directory of MP3s under a clips folder plus four TSVs — validated, invalidated, other, and reported — where the file you're in encodes the review status of every clip. The columns run client ID, path, sentence, up votes, down votes, age, gender, accent, locale, segment. Read those again as a set. That is not a modelling schema. That is the audit trail of a crowdsourcing pipeline. The vote columns and the demographic fields exist because Common Voice's mission is accent and language coverage, so the metadata that matters is who spoke and whether the crowd vouched for them.



Corn

Nobody training a model needs the down-vote count.



Herman

Nobody training a model needs it, and every linguist studying accent coverage needs it desperately. Corpus twenty point, end of twenty twenty-four, was about eighty-five gigabytes and around thirty-six hundred recorded hours, about twenty-six hundred of them validated, all CC-zero. And each release is a full dump — though they've since added Delta Segments, which are just the clips since the last release, about nine hundred megabytes for that one.



Corn

And the other one is the cautionary tale.



Herman

DeepSpeech. Three-column CSV, exactly: wav filename, wav filesize, transcript. Sixteen kilohertz mono WAV. And the filesize column is my favourite single design detail in any of these formats, because it exists purely to group similar-length audio together for efficient batching. That's it. That's the whole reason it's a column. A format shaped end to end by one trainer's batching strategy, and when DeepSpeech died, the format died with it.



Corn

A perfectly good column, and its only surviving relative is a training note.



Herman

So Daniel's instinct was right on both counts — LJSpeech layouts and NeMo manifests are the two named speech formats worth knowing, and there isn't a hidden third. Common Voice and DeepSpeech are the closest others, and both are tied to a single project. Common Voice survives because the project survives. DeepSpeech didn't.



Corn

Which is the fossil thesis in miniature. Let's cross to images, where the benchmark is the shaping force instead of the trainer. What changes?



Herman

What changes is that the format stops being about feeding a trainer efficiently and starts being about being scored fairly. COCO, Pascal VOC and YOLO differ on all three of our axes — file organization, geometry, extra flags — and each difference traces back to who the format answered to. COCO is one JSON file per split. A single object with three required top-level keys: images, annotations, categories. Each annotation carries an image ID, a category ID, a bounding box, an area, a segmentation, and an iscrowd flag. And the box is written as x, y, width, height — top-left corner plus size, explicitly not two corner pairs. The COCO API is precise about what that means: a box of zero, zero, one, one encloses the first pixel. Corners sit on pixel edges, not on pixels.



Corn

Meaning COCO thinks of an image as a continuous plane with pixels as cells inside it, and VOC thinks of it as a grid of numbered slots.



Herman

That's exactly the difference, and COCO's is the mathematically cleaner one, which is probably why it's also the one people get wrong when converting. Then the two traps. The twenty seventeen detection set has eighty classes, but the category IDs run one to ninety — they're a subset of the ninety-one stuff-and-thing categories from the original design, and eleven of them are just missing. Microsoft's own format notes warn that IDs should be consecutive integers starting from one, because many tools, CVAT and the official COCO API among them, will simply fail otherwise. You cannot tidy the numbering. The gap is load-bearing.



Corn

Twenty twenty-six's annotation tools, still respecting a hole left in twenty fourteen.



Herman

History frozen into the present, in the most literal possible form — an ID range with eleven holes in it. And the segmentation side is where COCO gets clever, which I know is the part where I lose you.



Corn

Herman, I once watched you explain run-length encoding at a Seder.



Herman

It was relevant! Briefly, though — COCO's mask encoding, RLE. Uncompressed, it's run lengths over the mask read column-major, starting with a run of zeros. Compressed, it's a string using six bits per character, ASCII forty-eight to one hundred eleven, and here's the elegant part: each stored value is the difference from the count two places back, so a mask whose column profile changes slowly — which is most of them — stays tiny. The compressed size grows with roughly the square root of the object's area for simple shapes.



Corn

Cleverness that only pays off if you never have to debug it.



Herman

Which is a fair summary of my relationship with it, yes. Why COCO won, though — it's the only one of the three that natively carries per-instance segmentation masks alongside boxes, and keypoints, captions, panoptic, DensePose, all in one schema. It arrived at ECCV twenty fourteen with two and a half million labelled instances across three hundred twenty-eight thousand images and ninety-one proposed categories, and it displaced VOC as the detection benchmark. Then the JSON became the lingua franca — LVIS follows the layout, SA-1B stores its masks in COCO RLE, Cityscapes ships converters to it. Not because the JSON is beautiful. Because everyone already parsed it.



Corn

VOC, by contrast, is one XML file per image.



Herman

Per image, with a filename, a size block, and one object block per instance: name, pose, truncated, difficult, and the bounding box as xmin, ymin, xmax, ymax. And the coordinates are one-based inclusive pixel indices — the evaluation code literally adds one to every width and height it computes. Segmentation lives somewhere else entirely, as PNG label maps with a void value of two hundred fifty-five.



Corn

One-based. In a computer format. And the reason it survived its own off-by-one is the challenge itself.



Herman

That's the part I find moving, if a file format can be moving. VOC started in two thousand five with four classes and fifteen hundred seventy-eight images. By two thousand seven it was twenty classes, nearly ten thousand images, twenty-four thousand objects, and segmentation had been added. VOC twenty twelve train and val had eleven and a half thousand images. The Everingham papers state the aim outright: a standard dataset of images and annotation, and standard evaluation procedures, so algorithms could be compared fairly, year over year. It was the first format with a real evaluation protocol attached. The protocol made VOC the reference. The XML was just how the protocol shipped.



Corn

So the format won because the scoreboard won. YOLO is the third one, and it's the most austere.



Herman

YOLO TXT is one text file per image, no header, one row per object: class, x-centre, y-centre, width, height — all five numbers, and all of them normalized to between zero and one. Classes start at zero. An image with no objects in it needs no file at all. The example row in the Darknet readme is class one, x-centre 0.717, y-centre 0.396, width 0.216, height 0.147. Five floats. Done.



Corn

And that's the whole format. There's a names file mapping IDs to strings and a data file pointing at the train list, and Ultralytics adds a YAML with the class names, but the annotation itself is five numbers.



Herman

And why it won is the normalization. Roboflow's format page puts it plainly — the zero-to-one range makes the annotations easy to work with even after scaling or stretching the image. They survive any resize, which is exactly what YOLO's multi-scale training does constantly. It's also the cheapest format in existence to write and parse — no XML, no JSON, no schema. The price is that there's almost no metadata. No difficult flag, no iscrowd, no image size recorded anywhere, no polygons in detection mode. Five numbers per object and silence.



Corn

You trade an entire audit trail for resize-proofing.



Herman

You do, and most of the time the trade is worth it, because YOLO's loaders regenerate what they need. But now assemble the conversion problem. VOC is one-based inclusive. COCO and YOLO are zero-based. VOC gives corners, YOLO wants centres, COCO wants corner-plus-size. The standard VOC-to-YOLO formulas divide by image dimensions, and because of the base difference you must subtract one from the min values first — skip that and every box shifts by a pixel. Computer Vision Online has the best line about this whole territory: the same four numbers can describe different rectangles, and a conversion can quietly move every box.



Corn

Quietly is the operative word. A one-pixel shift doesn't crash anything.



Herman

Nothing crashes. Your metrics sag a little, you shrug, you move on, and your model is subtly worse forever. There's a worse one, too — Ultralytics' own VOC-to-YOLO converter skips objects flagged difficult, entirely. So converting a VOC dataset through the default path silently deletes every hard example you were probably training on. The hardest



Herman (continued)

objects in your dataset, gone, no error, no warning.



Corn

A model that never sees the difficult cases, congratulating itself on its clean data. Well, that covers the image formats. Dorothy, have you got something for us?

D

Dorothy

Hello dear, it is time for my grim fact.

D

Dorothy

The Pascal VOC challenge began in two thousand and five with four classes and one thousand five hundred and seventy eight images.

D

Dorothy

Now, did you ever get those socks back, sweetheart, or are they still in the machine?

D

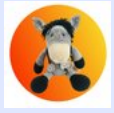
Dorothy

I have a casserole in, so ring me when you can. Bye bye, dear.



Corn

I'll ring her after the show.



Herman

Please do. Four classes, fifteen hundred images — and it grew into the reference protocol for a decade of computer vision. Mum has a gift for scale.



Corn

Everything we've described so far assumes the image is a given, and the format's job is to say where the boxes are. Three-D breaks that assumption entirely.



Herman

It does, and this is where the episode gets unresolved, because the conventions are forming right now. The clearest live example is a Hugging Face org that's been repackaging 3D scan collections as WebDatasets. The most downloaded one is Google Scanned Objects — just over a thousand household objects, forty-three tar shards, created this August, a few thousand downloads in the last month. Per sample, you don't get one file. You get a bundle: the original OBJ mesh, its material file, the diffuse texture as a PNG, a normalized self-contained GLB, five render thumbnails, and a JSON sidecar with metadata and processing stats. And the WebDataset convention is what holds it together: inside each tar, files that belong together share the same basename with the extensions stripped. Shards are numbered sequentially, typically around a gigabyte each, and the whole thing can run to terabytes.



Corn

The tar is the format and the contents are just... co-named.



Herman

Co-named and sequentially readable, which is the practical point — streaming a tar is three to ten times faster than random access to millions of small files, and when your samples are bundles rather than single images, that gap is the difference between a workable pipeline and one that spends its life seeking.



Corn

The part I find more interesting is the normalization recipe. Why does the card have to spell out geometry rules that a two-D format never would?



Herman

Because two-D formats are annotation-centric and three-D is asset-centric. In images, the pixel grid fixes everything — origin, orientation, units, scale, they're all given before any format says a word. In 3D there is no given. Every mesh arrives in its own coordinate frame, at its own scale, in its own units — one scan might be in centimetres, another



Herman (continued)

normalized by a completely different convention. So the dataset card has to become the format. This one spells it out: copy the texture next to the OBJ so the material file resolves, load with trimesh without merging anything, translate the bounding-box centre to the origin, uniformly scale so the longest edge is one — a unit cube — preserve the UVs, and export a self-contained GLB with the texture embedded. All of it produced deterministically, with a pinned trimesh version, and every transform logged to a JSONL file.



Corn

Pinning the library version. Because the format isn't the file, it's the file plus the exact software that made it.



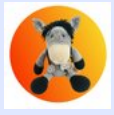
Herman

In 3D right now, yes. The recipe is the schema. And there's a detail that shows how thin the consensus really is — the card documents sixteen objects from the source collection that were excluded, because their original authors used a different file structure, collada files instead of OBJ, optional textures, variable thumbnail counts, and different licences. Sixteen objects dropped for non-conformance. In two-D, nothing like that happens. COCO JSON takes any image. In 3D, the objects that don't fit the recipe just fall off the dataset.



Corn

The licence exclusion is at least defensible. The file-structure exclusion is a confession.



Herman

It's an admission that the convention covers only what it covers. And the newest twist is that this org's most recent work isn't meshes at all — in the last couple of days they've been publishing latent packs, named for the TRELLIS-2 model, precomputed latents for the Smithsonian and NASA corpora, plus prep shards and a dry run that appeared just today. The Google Scanned Objects card names the intended use as direct input to TRELLIS-2 fine-tuning.



Corn

So 3D may skip the annotated-asset stage entirely.



Herman

That's what it looks like from here. Image diffusion went through the same split — raw images for some purposes, precomputed VAE latents for the training run — and 3D appears to be heading there faster, possibly skipping the COCO-equivalent stage on the way. Which would make 3D's first real standard a tensor layout rather than a file format. Two-D has a de facto schema and inherited its container. Three-D has a de facto container, the tar shard, and no schema at all — just recipes, written per project into the card.



Corn

So the answer to Daniel's last question, why each won its corner, closes the loop we opened with. LJSpeech won because torchaudio shipped a loader. YOLO won because Darknet needed floats that survive resizing. DeepSpeech's csv died with its trainer. VOC won on its scoreboard, not its XML. COCO's eleven missing IDs are still respected by tools written last month.



Herman

And nothing here was designed as a standard. Every one of them is a working solution that outlived its problem. The weird part is how well that process works — the fossil record is arguably a better standards body than most standards bodies, because the failure mode of a bad fossil is that it just disappears, quietly, like DeepSpeech's filesize column.



Hilbert

Simple is right, that format.



Hilbert

Corn keeps saying pipe-delimited like it's a scandal. It's not a scandal. The delimiter is the easy half. The third column is where the work is, and everyone treats it like a tidy copy. It isn't. It's the whole pronunciation contract.



Corn

The expansion column.



Hilbert

That one. I kept a list of the edge cases, on paper, because the tools kept disagreeing. Nineteen eighty-four is fine in a date and wrong in a page count. Doctor versus Drive after Dr. — context, every time, no exceptions. I settled that years ago. The shop that printed my labels had a man who did the expansions by hand before the software existed, and he never once asked what a number meant. He just knew. I learned more from arguing with him than from any tool since.



Herman

You kept a handwritten normalization list?



Hilbert

Still have it. It's not a script. It's a list. Hold on — Herman, you're clipping on the last word of every sentence, I'll ride the gate after this.



Hilbert

Anyway. The point of column three is that somebody decided. Most people never notice anyone decided.



Corn

Your man at the printers decided everything, then. One brain, no version number.



Hilbert

He did, and he'd have been horrified by a JSON manifest. And yes, one brain. I checked his work twice. He was right both times, which I still don't enjoy saying.



Herman

Two checks against a man doing numbers by ear, and he won both.



Hilbert

You want the expansion done properly, you find out who your equivalent of him is and you keep them.



Corn

Whether that person is a man at a print shop or a pinned trimesh version — every format we've shown Daniel today has one, buried somewhere. The open question I'd leave him with is the 3D fork. If the first real 3D standard turns out to be a tensor layout for precomputed latents rather than a file format at all, does that make the whole named-format era a two-D-and-audio phenomenon — or does every modality eventually converge on latents and skip the annotation schema?



Herman

And the practical residue, the thing I'd actually carry out of this: the base matrix. One-based versus zero, inclusive versus exclusive, corners versus centres. That one silent off-by-one that shifts every box in a VOC-to-YOLO conversion is the same class of error that ruins transcript alignment in ASR work, which is Daniel's actual day job with this data. Know which convention you're standing in before you trust a number that looks fine.



Corn

Thanks as always to Hilbert Flumingtop at the desk, for the producing and the gate-riding. For more along these lines, there's episode three, Safetensors or something else; episode nine, Benchmarking Custom ASR Tools - Beyond The WER; and episode seven, Building Custom ASR Tools. This has been My Weird Prompts — no question too obscure, no rabbit hole too deep.



Herman

If the show's been worth your time, a review wherever you listen helps more people find us. See you tomorrow, Corn.



Corn

See you tomorrow.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **The LJ Speech Dataset, official page , retrieved 2026-11-20. [primary]**
   https://keithito.com/LJ-Speech-Dataset/

2. **NVIDIA NeMo-Speech, Datasets (manifest format), retrieved 2026-11-20 . [primary]**
   https://docs.nvidia.com/nemo/speech/latest/asr/datasets.html

3. **NVIDIA NeMo Framework User Guide, Datasets, retrieved 2026-11-20 . [primary]**
   https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/asr/datasets.html

4. **NeMo-Speech audio-to-audio datasets, retrieved 2026-11-20 . [primary]**
   https://docs.nvidia.com/nemo/speech/latest/audio/datasets.html

5. **NeMo Curator, Dataset Manifests and Ingest, retrieved 2026-11-20 . [primary]**
   https://docs.nvidia.com/nemo/curator/about/concepts/audio/manifests-ingest

6. **nvidia/Granary dataset card (NeMo JSONL manifests at scale), retrieved 2026-11-20 . [primary]**
   https://huggingface.co/datasets/nvidia/Granary

7. **torchaudio LJSPEECH loader source, retrieved 2026-11-20 . [primary]**
   https://docs.pytorch.org/audio/stable/_modules/torchaudio/datasets/ljspeech.html

8. **Mozilla Common Voice datasets page (release table), retrieved 2026-11-20 . [primary]**
   https://commonvoice.mozilla.org/en/datasets

9. **PASCAL VOC official homepage , retrieved 2026-11-20. [primary]**
   https://www.robots.ox.ac.uk/~vgg/projects/pascal/VOC/

10. **COCO official data format page , retrieved 2026-11-20. [primary]**
    https://raw.githubusercontent.com/cocodataset/cocodataset.github.io/master/dataset/format-data.htm

11. **Microsoft COCO data format notes (ID numbering warning), retrieved 2026-11-20 . [primary]**
    https://github.com/microsoft/vision-datasets/blob/main/COCO_DATA_FORMAT.md

12. **COCO dataset homepage (1.5M instances, 80 object / 91 stuff categories) , retrieved 2026-11-20. [primary]**
    https://cocodataset.org/dataset/home.htm

13. **AlexeyAB Darknet README (obj.data, example label row) , retrieved 2026-11-20. [primary]**
    https://github.com/AlexeyAB/darknet/blob/Yolo_v3/README.md

14. **Ultralytics VOC config and XMLYOLO converter (skips `difficult`), retrieved 2026-11-20 . [primary]**
    https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/datasets/VOC.yaml

15. **hugging-3d/google-scanned-objects dataset card (WebDataset, GLB normalization), created 2026-08-17, retrieved 2026-11-20 . [primary]**
https://huggingface.co/datasets/hugging-3d/google-scanned-objects

16. **WebDataset README (tar conventions, shard naming, I/O speedup), retrieved 2026-11-20 . [primary]**
https://github.com/webdataset/webdataset

17. **Hugging Face Hub, WebDataset docs (~1 GB shards, multi-TB datasets), retrieved 2026-11-20 . [primary]**
https://huggingface.co/docs/hub/en/datasets-webdataset

18. **NVIDIA DALI WebDataset reader docs, retrieved 2026-11-20 . [primary]**
https://docs.nvidia.com/deeplearning/dali/user-guide/docs/operations/nvidia.dali.fn.readers.webdataset.html

19. **Keras CTC ASR example documenting LJSpeech metadata fields, retrieved 2026-11-20.**
https://github.com/keras-team/keras-io/blob/master/examples/audio/ctc_asr.py

20. **State Library of Queensland, Datasets in detail, Andrei Maberley, 2023-11-17, retrieved 2026-11-20.**
https://www.slq.qld.gov.au/blog/datasets-detail

21. **Mozilla Discourse on Common Voice TSV semantics, 2021-06-12, retrieved 2026-11-20.**
https://discourse.mozilla.org/t/where-is-the-documentation-regarding-mozilla-common-voice-database/81476

22. **SpeechBrain Common Voice prep script, retrieved 2026-11-20.**
https://github.com/speechbrain/speechbrain/blob/develop/recipes/CommonVoice/common_voice_prepare.py

23. **DeepSpeech Playbook, Data Formatting, retrieved 2026-11-20.**
https://mozilla.github.io/deepspeech-playbook/DATA_FORMATTING.html

24. **Annotation Formats (COCO/VOC/YOLO comparison, RLE details), retrieved 2026-11-20.**
https://computervisiononline.com/explains/annotation-formats/

25. **PASCAL VOC history and purpose, retrieved 2026-11-20.**
https://computervisiononline.com/explains/pascal-voc/