Video Learning

Chris Hoang
2025-02-06

General deep learning process

Dataloading

1. Load raw data
2. Preprocess data
3. Gather data samples into batches

Model training

1. Load data batch onto GPU
2. Perform model forward pass and loss computation
3. Perform backpropagation to update model parameters

Dataloading

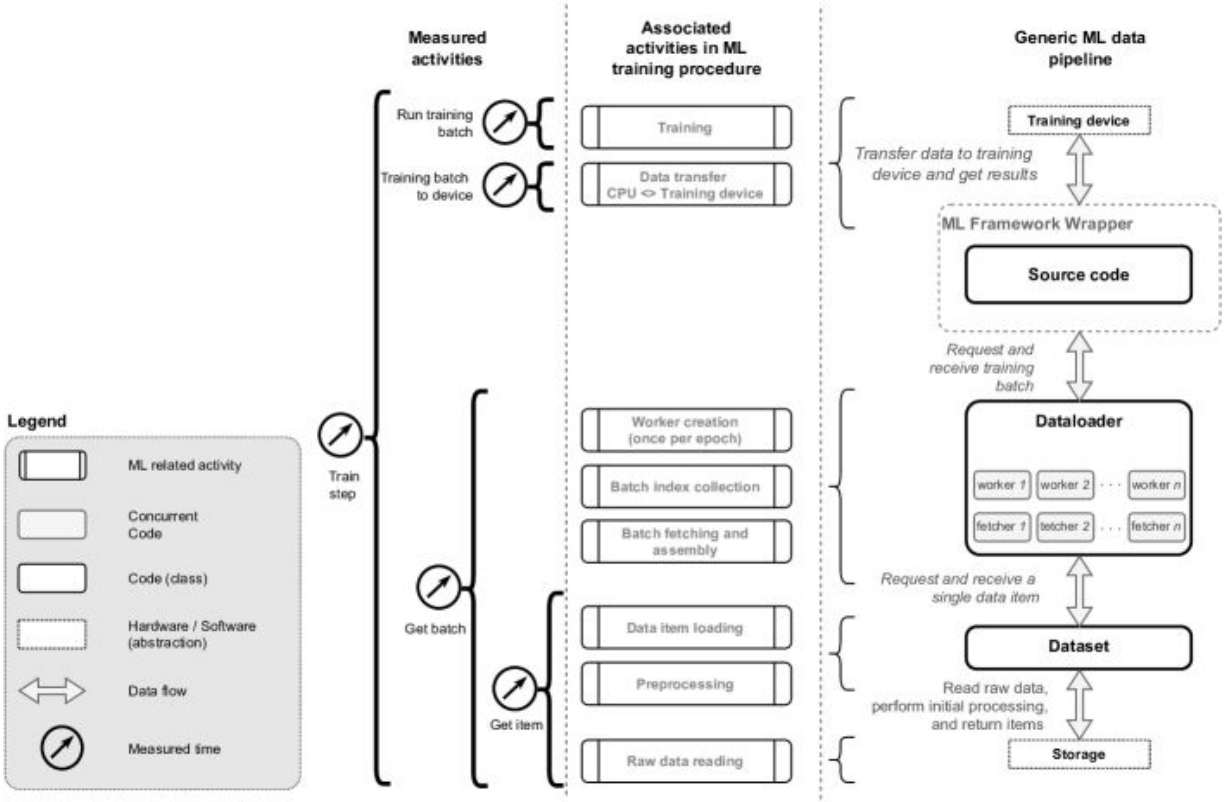
Data workers run on CPU cores

Runs asynchronously while model training is done on GPU

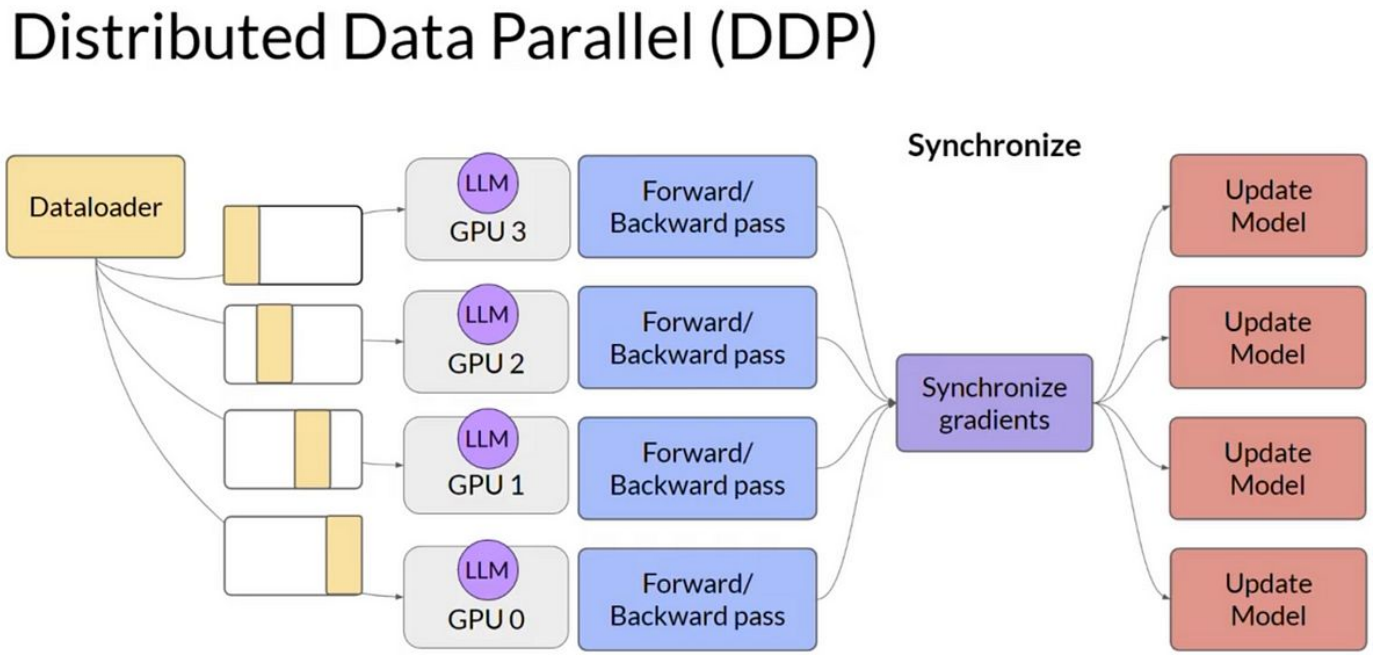
Dataloading

Data workers run on CPU cores

Runs asynchronously while model training is done on GPU



Distributed training



Distributed training

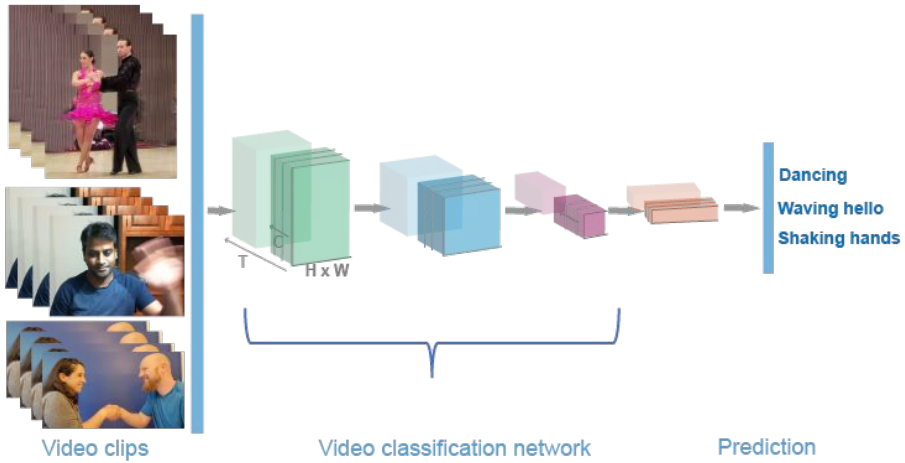
```python
def init_distributed():

    if dist.is_available() and dist.is_initialized():
        return dist.get_world_size(), dist.get_rank()

    rank, world_size = None, None

    if (rank is None) or (world_size is None):
        try:
            if "SLURM_NNODES" in os.environ and "SLURM_PROCID" in os.environ and "SLURM_TASKS_PER_NODE" in os.environ:
                world_size = int(os.environ["SLURM_NNODES"]) * int(os.environ["SLURM_TASKS_PER_NODE"][0])
                rank = int(os.environ['SLURM_PROCID'])
            elif "WORLD_SIZE" in os.environ and "RANK" in os.environ:
                world_size = int(os.environ["WORLD_SIZE"])
                rank = int(os.environ['RANK'])
            gpu = rank % torch.cuda.device_count()
            if ("MASTER_ADDR" in os.environ and "MASTER_PORT" in os.environ):
                dist_url = "tcp://{}:{}".format(os.environ['MASTER_ADDR'], os.environ['MASTER_PORT'])
            else:
                dist_url="tcp://localhost:8965"
        except Exception:
            log.info('SLURM vars not set (distributed training not available)')
            world_size, rank = 1, 0
            return world_size, rank

    try:
        log.info('| distributed init (rank {}): {}, gpu {}, world_size {}'.format(rank, dist_url, gpu, world_size))
        dist.init_process_group(
            init_method=dist_url,
            backend='nccl',
            world_size=world_size,
            rank=rank)
        torch.cuda.set_device(gpu)
```

Videos

Sequences of image frames

Store key frames and “deltas” between key frames to achieve storage (lossy) compression

Enable learning temporal information, motion, object behavior, world models



Video datasets

Kinetics400: clips of human actions

Walking Tours: walking city tours

SomethingSomething: clips of human actions

Ego4D: egocentric daily life videos

BDD100K: driving dashcams

Waymo Open: driving camera videos

Video datasets sizes

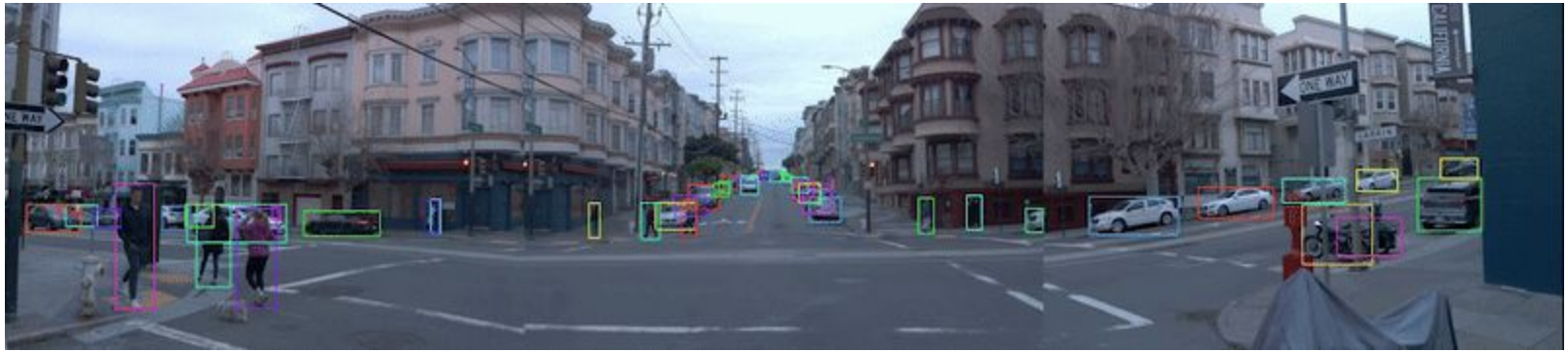
Dataset	Domain	Ego	Pre	Bal	Annot	Avg. Dur (sec)	Dur (hr)	#Videos	Frame resolution
Diverse Pretraining									
Kinetics-400 (Kay et al., 2017)	Actions	✗	✓	✓	**Class**	10.2	851	400	340 × 255
WebVid-2M (Bain et al., 2021)	Open	✗	✓	✗	**Weak**	18	13k	–	320 × 240
HowTo100M (Miech et al., 2019)	Instructions	✗	✓	✗	**Weak**	4	135k	–	–
Egocentric									
Epic-Kitchens (Damen et al., 2022)	Cooking	✓	✗	✗	**Loc.**	510	100	37	1920 × 1080
Ego-4D (Grauman et al., 2022)	Daily	✓	✗	✗	**Loc.**	1446	120	931	1920 × 1080
Meccano (Ragusa et al., 2023)	Industry	✓	✗	✗	**Loc.**	1247	849	20	1920 × 1080
Assembly-101 (Sener et al., 2022)	Assembly	✓	✗	✗	**Loc.**	426	167	362	1920 × 1080
ImageNet-aligned									
R2V2 (Gordon et al., 2020)	ImageNet	✗	✓	✓	**Class**	–	–	–	467 × 280
VideoNet (Parthasarathy et al., 2022)	ImageNet	✗	✓	✓	**Class**	10	3055	–	
Walking Tours (ours)	Urban	✓	✓	✗	**None**	5880	23	10	3840 × 2160

For reference, ImageNet is 1.3M images of size ~470x390

Example: WalkingTours

Epic-Kitchens

Example: Waymo Open



Working with video data

Storage - time tradeoff:

1. Videos are compressed stacks of images
2. It takes time to decode videos into data arrays

Frame sampling

1. How much time in-between frames?
2. What level of temporal granularity do you care about?

Frame sampling

Δt = 0 (0s) Δt = 15 (0.5s) Δt = 30 (1s) Δt = 45 (1.5s)

Video learning demo

Video Tokenization

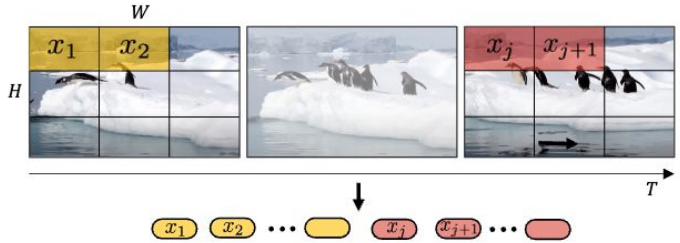


Figure 2: Uniform frame sampling: We simply sample n_t frames, and embed each 2D frame independently following ViT [18].

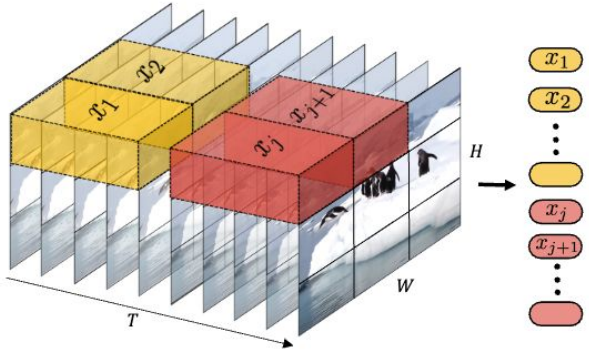
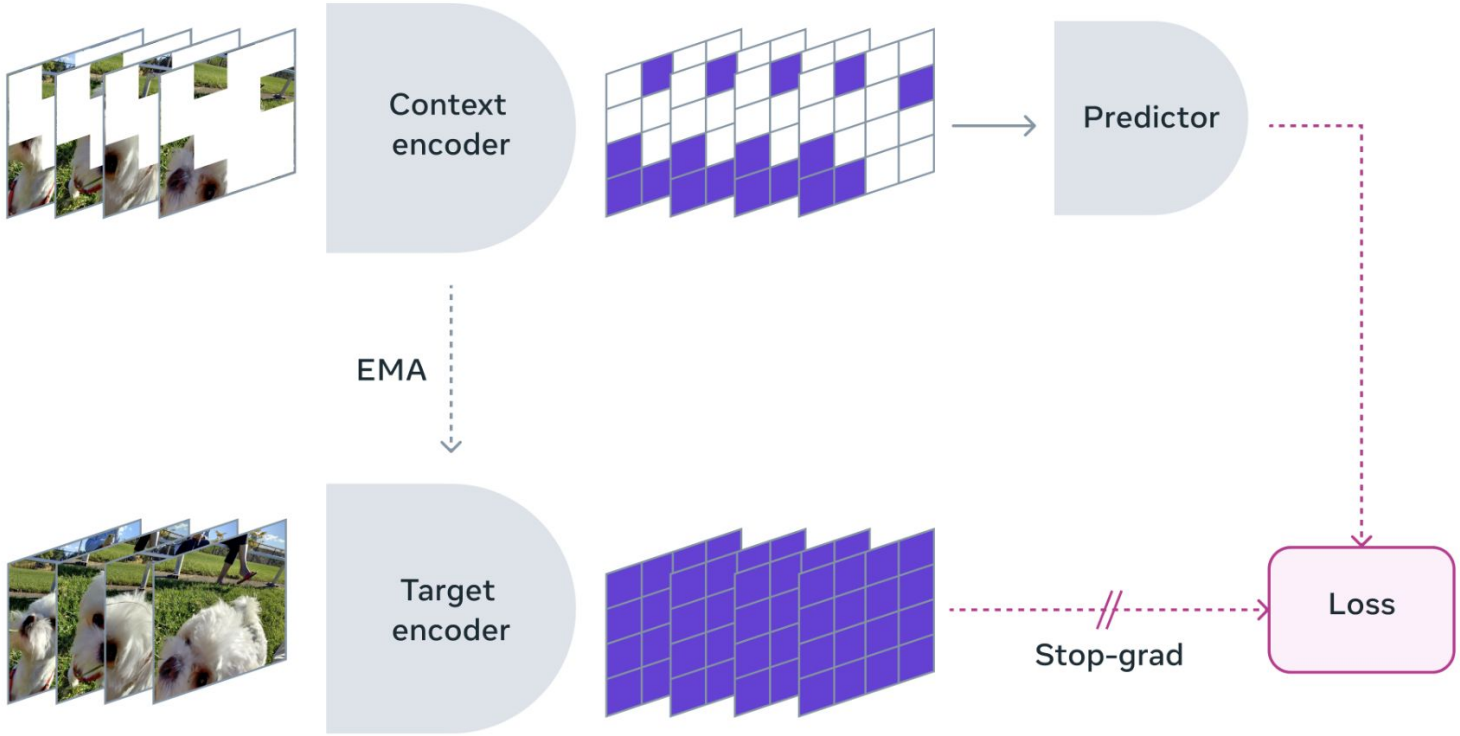


Figure 3: Tubelet embedding. We extract and linearly embed non-overlapping tubelets that span the spatio-temporal input volume.

V-JEPA


Context encoder
Predictor
EMA
Target encoder
Stop-grad
Loss