Training: Recommended Hyperparameters (Per-Device Batc

Overview

This document consolidates best-practice hyperparameters for your RWTH HPC SLURM framework to improve ASR and SER accuracy. Settings target a per-device batch size of 16 on H100 GPUs with mixed precision. Assumptions: distributed data parallel via torchrun, KenLM-enabled CTC decoding for ASR, and a frozen wav2vec2-style encoder with a small classifier head for SER.

Important note about your code

In your current training script, ASR uses per_device_train_batch_size = max(1, asr_batch_size // 2). Therefore, to achieve per-device=16 on ASR, set --asr_batch_size=32. For SER, --ser_batch_size maps directly to per-device.

Training: Recommended Hyperparameters (Per-Device Batc

ASR (CTC on Wav2Vec2/XLS-R + LoRA)

Core training

- Batch: set --asr_batch_size=32 → per-device=16 (your code halves it for padding headroom).
- Learning rate: 1e-4 (grid: 5e-5, 1e-4, 3e-4).
- Epochs / Early stop: 20 epochs, patience=5 on dev WER.
- Weight decay: 0.005.
- Warmup: 10% of total steps; constant or cosine after warmup both OK.
- Precision: fp16 (AMP) with grad_accum=4 if needed.
- Checkpointing: load_best_model_at_end = True.
- Optional: gradient_checkpointing = True (if VRAM tight).

Regularization (SpecAugment)

- Enable mild masking: mask_time_prob=0.05; mask_feature_prob=0.05–0.08.

LoRA

- Targets: q_proj,k_proj,v_proj,out_proj (+ optionally FFN). Start with r=8, alpha=32; try r=16 if VRAM allows.

Evaluation

- Report WER on a held-out dev set with the same text normalization as training.

ASR Decoding (pyctcdecode + KenLM)

- Start with beam_width=100, alpha=0.5, beta=1.0.
- Tune on dev: alpha ∈ {0.2, 0.5, 1.0}, beta ∈ {0.5, 1.0, 2.0}; keep the best pair.
- Train the KenLM n-gram on in-domain text; use .bin for fast load.
- (Optional) Shallow-fusion neural LM: rescore beams with a small LM; sweep LM weight 0.2–0.8.

Training: Recommended Hyperparameters (Per-Device Batc

SER (Wav2Vec2 encoder frozen + small MLP head + LoRA)

Core training

- Batch: --ser_batch_size=16 (per-device).
- Learning rate: 1e-4 (grid: 5e-5, 1e-4, 2e-4).
- Epochs / Early stop: 25 epochs, patience=5 on dev F1.
- Dropout: 0.3 (try 0.2–0.5).
- Weight decay: 0.01 (0.005–0.01 both reasonable).
- Schedule: linear with 10% warmup.
- Loss: class-weighted cross entropy (you already compute weights).
- Backbone: keep frozen initially; consider unfreezing the last 1–2 transformer blocks after 5–10 epochs if the dev metric plateaus.

LoRA

- Targets: attention and FFN blocks in the encoder + classifier head trainable. Start with r=8, alpha=16; test r=16 on more VRAM.

Inference

- If using sliding windows over long audio, prefer logit-averaging or temperature-scaled softmax averaging across windows to reduce overconfident spikes.

DDP & Cluster notes (H100 on CLAIX-2023)

- Per-device 16 + grad_accum=4 → effective batch per node = 64 × (#GPUs).
- Use NCCL and set NCCL_SOCKET_IFNAME to the InfiniBand device (e.g., ib0) for multi-node.
- Keep srun --ntasks-per-node=1 and torchrun --nproc_per_node=<GPUs>; let torchrun spawn one worker per GPU.

Training: Recommended Hyperparameters (Per-Device Batc

Quick CLI presets (drop-in)

ASR

```
  --phase asr \
  --asr_batch_size 32 \
  --asr_learning_rate 1e-4 \
  --asr_epochs 20 --asr_patience 5
```

SER

```
  --phase ser \
  --ser_batch_size 16 \
  --ser_learning_rate 1e-4 \
  --ser_epochs 25 --ser_patience 5 --ser_dropout 0.3
```

Decoder (dev-set search)

```
  beam_width: 50, 100
  alpha: 0.2, 0.5, 1.0
  beta: 0.5, 1.0, 2.0
```

References (selected)

[1] Hugging Face: Fine-tune Wav2Vec2 for ASR (LR~1e-4, warmup, etc.).
[2] Hugging Face: Fine-tune XLS-R for multilingual ASR (low-resource recipe & masking).
[3] pyctcdecode (KenLM integration and beam search with alpha/beta).
[4] NVIDIA NeMo: KenLM training & decoding integration guidance.
[5] Chen & Rudnicky (2021, 2023v3): Fine-tuning Wav2Vec2 for SER (V-FT, TAPT/P-TAPT).