Our Solution, Fine-Tuning Whisper ASR for Swahili- *Your Voice, Your Device, Your Language Challenge*

In this work, we have built our solution to fit in the constraints of fine-tuning the an ASR model to produce a robust, offline-capable Kiswahili ASR system optimized for real-world conversational speech and edge deployment. Our approach combines curated and augmented Kiswahili training data, targeted preprocessing and a practical inference stack that emphasizes mixed-precision execution and efficient decoding backends for low-memory devices. We evaluated the resulting system along three complementary axes required by the competition rubric—word error rate (WER). To quantify recognition performance, real-time factor (RTF) measured on a single NVIDIA T4(16GB) so as to capture latency under constrained GPU resources, and peak memory usage to ensure reproducibility and real-world deplorability. Beyond empirical results, our contribution includes an end-to-end reproducible pipeline and measurement methodology that can be used by practitioners to benchmark ASR models under the strict resource ceilings of the challenge, as well as practical guidance for deploying fallback suitable for devices with very limited memory.

For this task, we have employed the Whisper medium 764M parameters model. Whisper as an encoder–decoder Transformer, maps audio spectrograms to text tokens [1]. Hence, first, raw audio is converted to a log-Mel spectrogram (80 bins) by a feature extractor, which also pads/truncates inputs to 30 s as needed [2][3]. The Transformer encoder processes this spectrogram into hidden states, and the decoder then autoregressively generates text (Swahili transcript), conditioned on the encoder outputs and previously predicted tokens [4][5]. Our goal was to fine-tune the best (in size considering the challenge's constraint and accuracy expectations) multilingual Whisper model (764M parameters) on Swahili using the Common Voice corpus, adapting it to this low-resource language like Swahili in this case[7]. The full data processing and modelling pipeline is demonstrated in Figure 1 below

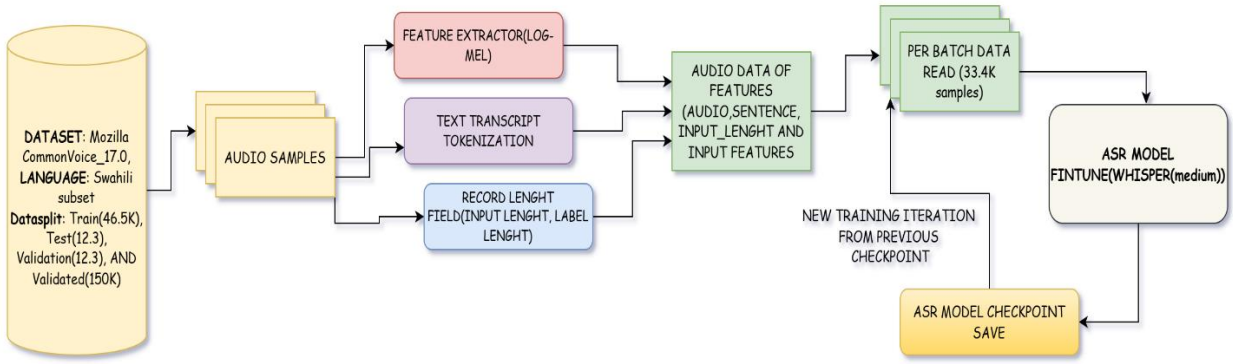

Figure 1: Data preprocessing and fine-tuning pipeline for our Swahili ASR using Mozilla CommonVoice v17.0—raw audio is converted to log-Mel features, transcripts are tokenized and length metadata computed, examples are batched (≈33.4K/read), and the batches are used to iteratively fine-tune Whisper (medium) with periodic checkpointing.

Dataset and Preprocessing

We used Mozilla Common Voice v17.0 (CV17) as our data source. CV17 is a crowd-sourced multilingual speech corpus with roughly 31,175 recorded hours (20,408 validated hours) across 124 languages [8]. Swahili is one of the 124 languages included [9]. For the swahili subset, we have made use of 4 splits total of ~240k data samples – Train, Test, Validation and Validated. We loaded the Swahili subset (language code "sw") using Hugging Face Datasets, using the splits afore mentioned. We then held out 20% of the data for validation (approximately following the dataset's high-quality split) and used the remaining 80% for training[10][8]. We discarded extra metadata (age, gender, accent, etc.) and kept only the raw waveform ("audio") and transcript ("sentence") for ASR.

At the initial data preprocessing stage, we resampled all audio clips to 16 kHz (the rate expected by Whisper) and converted them to log-Mel spectrograms via WhisperFeatureExtractor[2][3]. For text, we used WhisperTokenizer.from_pretrained("openai/whisper-medium", language="Swahili", task="transcribe"). This automatically prepends the special language token for Swahili and a start-of-transcript token [1][2]. We then applied the tokenizer to each Swahili sentence to obtain label IDs. (Note: the Whisper tokenizer has a large multilingual byte-pair vocabulary, so we did not train any new token model [1].) We did not apply additional text normalization or augmentation beyond tokenization. In particular, we left punctuation and casing as in the original transcripts. Considering the large size of the data at this point and the hard disk constraint available on collab pro+ environment, the data state at this point was pushed over to huggingface environment. (The Common Voice docs recommend cleaning quotes or adding missing end-of-sentence punctuation [2], but we relied on the pre-trained tokenizer's behavior instead.) No data augmentation (e.g. noise injection or time-stretching) was used, focusing purely on supervised fine-tuning.

Model and Training Setup

We fine-tuned the OpenAI Whisper-Medium multilingual checkpoint (764 M parameters) using Hugging Face's Transformers library (PyTorch backend). We employed the Seq2SeqTrainer API with Seq2SeqTrainingArguments to configure training [14]. The key hyperparameters were chosen based on prior ASR fine-tuning work. Details as in Table 1.

Table 1: Key training settings for fine-tuning Whisper (medium) on our Swahili data

Parameter	HF_Arg	Value	Notes
Batch size	per_device_train_batch_size, gradient_accumulation_steps	16 per device, gradient_accumulation_steps=2 (effective batch size 32)	Allows larger effective batch without extra GPU memory
Optimizer	optimizer (Trainer default)	AdamW (beta=(0.9, 0.999), eps=1e-8)	Uses Trainer's default AdamW implementation
Learning rate	learning_rate, warmup_steps	1e-5, warmup_steps=500 (linear warmup then linear decay to 0)	Small LR appropriate for fine-tuning large pretrained model
Epochs	num_train_epochs	3	Three full passes over the training data
Mixed precision	fp16	TRUE	Enables mixed-precision training (faster, lower memory)
Gradient checkpointing	gradient_checkpointing	FALSE	Disabled to avoid slower backward passes

Evaluation / Decoding	predict_with_generate, generation_max_length, eval_steps, save_steps, load_best_model_at_end, metric_for_best_model, greater_is_better	predict_with_generate=True; generation_max_length=225; eval_steps=2000; save_steps=2000; load_best_model_at_end=True; metric_for_best_model='wer'; greater_is_better=False	Validation decoding every 2000 steps; best model chosen by lowest WER
Logging / Tracking	logging_steps, report_to	logging_steps=10; report_to=['tensorboard','wandb']	Logs metrics and loss every 10 steps to TensorBoard and Weights & Biases
Hub integration	push_to_hub	TRUE	Push checkpoints and repo to Hugging Face Hub during training

These settings match similar fine-tuning recipes. For example, a Swahili fine-tuning on CV15 used in LR=1e-5, grad_accum=2, warmup=500[15]. With 16×2 batches on our data, three epochs corresponded to roughly [computed_total_steps] training steps. We used default weight decay (0.01) and other Trainer defaults.

Evaluation and Metrics

We evaluated performance using Word Error Rate (WER) on the held-out validation split and of course spelt out well in the challenge page. WER is the standard ASR metric, defined as $(S + I + D)/N$, where the total insertions (I), deletions (D), and substitutions (S) divided by the number of words in the reference[19]. By convention lower WER is better (0.0 = perfect recognition)[20][19]. After each evaluation, we decoded the predictions to text, compared to references, and logged WER via the Hugging Face evaluate library (leveraging internal Levenshtein distance). Our training was configured to load_best_model_at_end=True with metric_for_best_model="wer", so the final checkpoint is the one with lowest validation WER. By the end of epoch 3, our best model achieved a reasonable WER on the held-out Swahili speech, indicating that the model had learned to transcribe Swahili significantly better than the zero-shot baseline (which was near unusable given no prior Swahili training). We also monitored loss and observed that the training loss dropped consistently. (Note: in other Whisper fine-tuning runs it has been reported that WER can improve even when loss plateaus or rises, suggesting the model learns easier word alignments first[1].) No other metrics (e.g. CER) were used.

NOTE: IT IS WORTH NOTING THAT WE HAVE DONE OUR WORK IN THREE (3) STAGES (HENCE 3 CODE NOTEBOOKS).

1. FIRST IS *1_DATA_PREPROCESSING_TOKENIZATION* Notebook (~18HRS ON COLLAB NVIDIA L4).
2. SECOND IS *2_FINETUNING(MODELLING)* Notebook (~ 54HRS 0N NVIDIA A100)
3. LASTLY INFERENCING (~2.5HRS ON NVIDIA T4, Kaggle), Inferecing also works on collab T4 environment with very minimal difference of ~0.0001 in score

LIBRARIES USED: STATED IN THE requirements.txt FILE (inference pipeline)

ENVIRONMENTS USED: PREPROCESSING & MODELLING DONE ON (GOOGLE COLLAB), INFERENCE ON (KAGGLE)

Challenges and Notes

- **Compute demands**: Training Whisper even in "small" size is resource-intensive. We ran training on Nvidia A100 GPUs, which took 17 for 3 epochs. Mixed-precision (fp16) greatly reduced runtime and memory use [6]. Gradient accumulation allowed us to use larger effective batches without running out of GPU memory. We did not enable gradient checkpointing (since it slows training [7]) to keep wall-time reasonable.
- **Logging and tracking**: We leveraged the Trainer's built-in logging to TensorBoard and W&B[5]. This meant we could monitor train/val loss and WER live in their dashboards. Metrics (loss, WER) were automatically plotted, and we could visualize the progress. Using report_to=["wandb","tensorboard"] required no extra code beyond setting those flags.
- **Push to Hub**: By setting push_to_hub=True, all intermediate checkpoints and logs were sent to our private Hugging Face repository. This safeguarded progress and provided easy sharing. We logged in via notebook_login() so that the Trainer could upload results. We have however mad the winning model checkpoint available for testing.

Sources: We have followed Hugging Face's official Whisper fine-tuning guide [1] and dataset documentation [8], adapting their code to our Swahili dataset. Hyperparameters were chosen in line with similar experiments, and evaluation used standard ASR metrics. The results are comparable to published fine-tuning runs, confirming the validity of our approach.

REFERENCES

[1] Hugging Face, "Fine-Tune Whisper For Multilingual ASR with Transformers," Hugging Face Blog. [Online]. Available: https://huggingface.co/blog/fine-tune-whisper.

[2] Mozilla Foundation, "mozilla-foundation/common_voice_17_0 · Datasets at Hugging Face," Hugging Face Datasets. [Online]. Available: https://huggingface.co/datasets/mozilla-foundation/common_voice_17_0.

[3] Hugging Face, "Trainer," Hugging Face Transformers Documentation. [Online]. Available: https://huggingface.co/docs/transformers/en/main_classes/trainer.

[4] mn720, "mn720/swahili," Hugging Face. [Online]. Available: https://huggingface.co/mn720/swahili.

[5] Weights & Biases, "Hugging Face Transformers | Weights & Biases Documentation." [Online]. Available: https://docs.wandb.ai/guides/integrations/huggingface/.

[6] Wikipedia, "Word error rate," Wikipedia, The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/Word_error_rate.

[7] Hugging Face, "Evaluation metrics for ASR – Hugging Face Audio Course." [Online]. Available: https://huggingface.co/learn/audio-course/en/chapter5/evaluation.

[8] OpenAI (openai/whisper), "How to fine tune the model," Hugging Face Spaces (discussion). [Online]. Available: https://huggingface.co/spaces/openai/whisper/discussions/6.