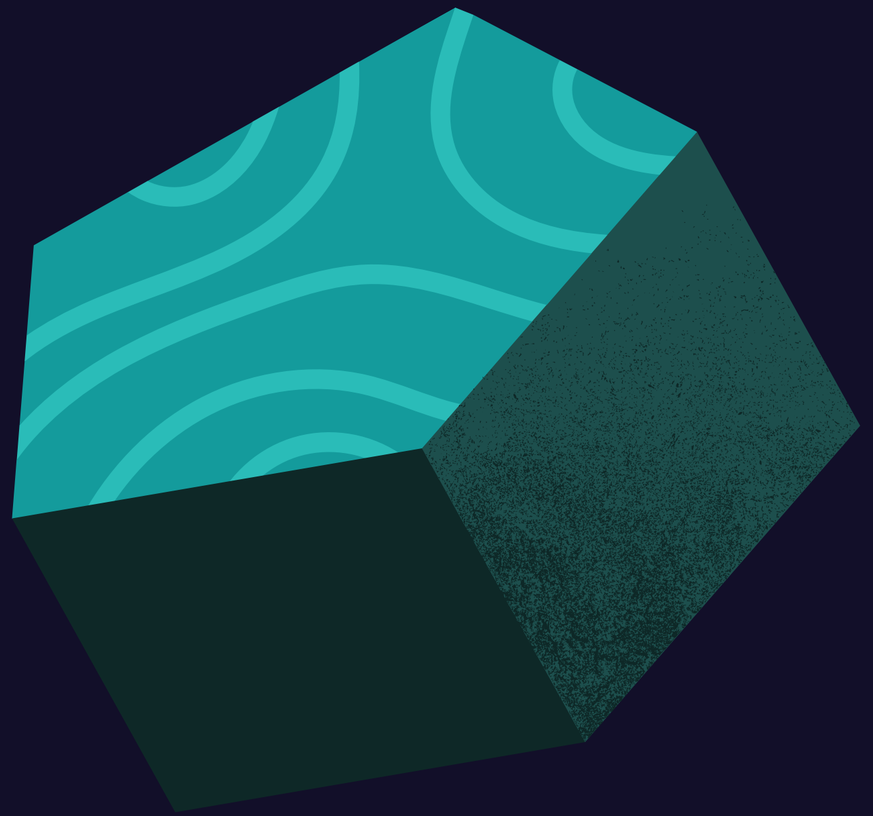
ASR for Egyptian Dialects

MTC-AIC2

Today's Agenda

1. Introduction
2. Wav2Vec2
3. Turning to the FAdam Optimizer
4. Knowledge Distillation Model
5. Conformer-CTC
6. Data Augmentations and Tokenizer
7. Results



Omar Ismail

Mohamed Motawie

Mohamed Tamer

Introduction

Welcome to our detailed presentation on the MTC-Competition. In this competition, we embarked on a journey to enhance the performance of advanced speech recognition models, focusing on Wav2Vec2 and Conformer-CTC.

Our goal was to push the boundaries of automatic speech recognition (ASR) by overcoming various technical challenges and implementing innovative solutions.

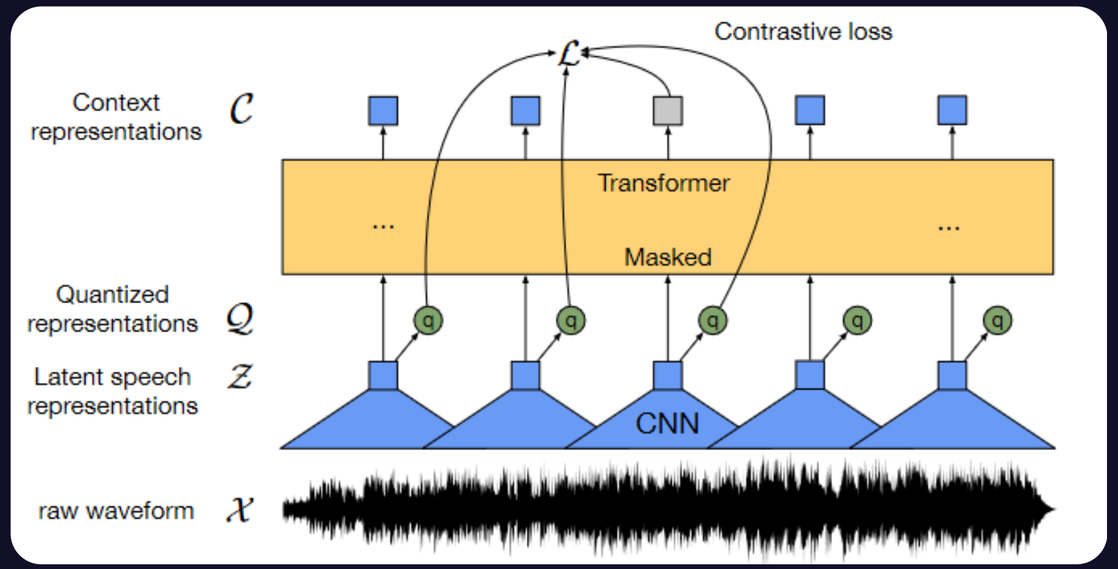
Throughout this presentation, we will take you through the significant obstacles we encountered and the strategies we employed to address them. From the instability issues in pre-training Wav2Vec2 to the complexities of implementing effective tokenization techniques and optimizing training procedures, we faced numerous hurdles that required creative problem-solving and persistent experimentation.

We will also delve into the transformative impact of the FAdam optimizer, the challenges of knowledge distillation, and our ultimate pivot to the Conformer-CTC model.

Wav2Vec2

Our journey began with high hopes pinned on Wav2Vec2, a model known for its cutting-edge performance in speech recognition tasks.

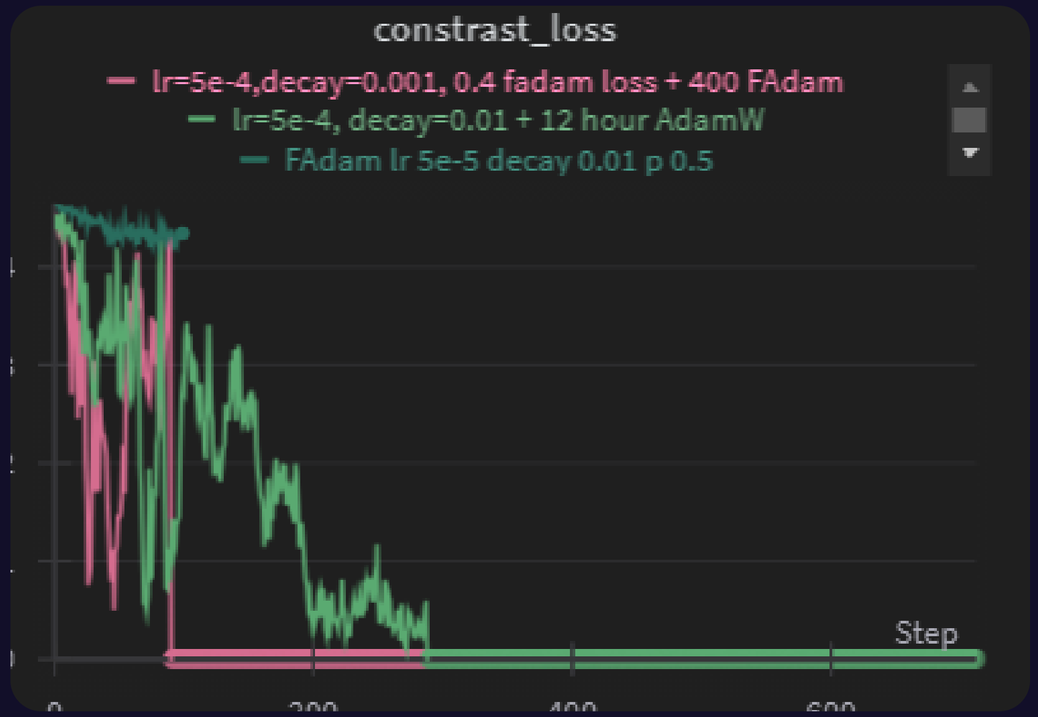
- **Self-Supervised Learning**: Learns speech representations without the need for transcriptions during pre-training.
- **Transformer Architecture**: Uses a transformer-based model to process speech data.
- **Quantization**: Applies quantization to the continuous speech representations, aiding in learning discrete latent speech units.
- **Fine-Tuning**: The pre-trained model can be fine-tuned on labeled data for specific ASR tasks.



Instability in Pre-Training Wav2Vec2

Gradient Problems:
The instability manifested in erratic gradients, which are essential for the optimization process. Erratic gradients can disrupt the learning process, causing the model to fail in learning meaningful patterns from the data.

NaN Values:
These values appeared during training, indicating numerical instability. This could be due to issues such as too high learning rates or poor initialization of model parameters.



FAdam Optimizer

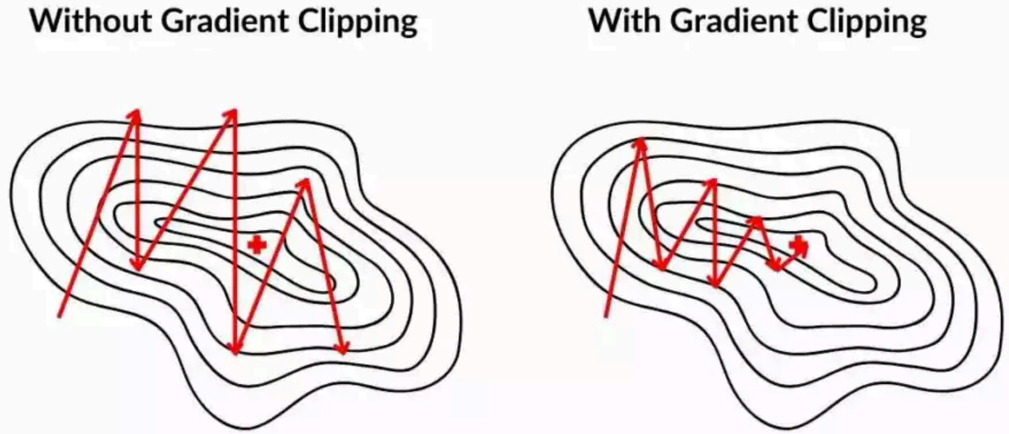
Adaptive Epsilon and Gradient Clipping: Dynamically adjusts epsilon based on training progress for optimal performance and employs gradient clipping to prevent excessively large gradients

Empirical Fisher Information: Uses the diagonal empirical (FIM), capturing variances but not covariances, for computational efficiency.

Enhanced Weight Decay: Applys weight decay based on the FIM, which ensures that the regularization is consistent with the underlying geometry of the parameter space..

LibriSpeech WERs	dev	dev-other	test	test-other	avg
Adam (w2v-BERT paper [47])	1.30	2.60	1.40	2.70	2.00
Adam	1.30	2.54	1.33	2.59	1.93
FAdam	1.29	2.49	1.34	2.49	**1.89**

Table 1: LibriSpeech WERs



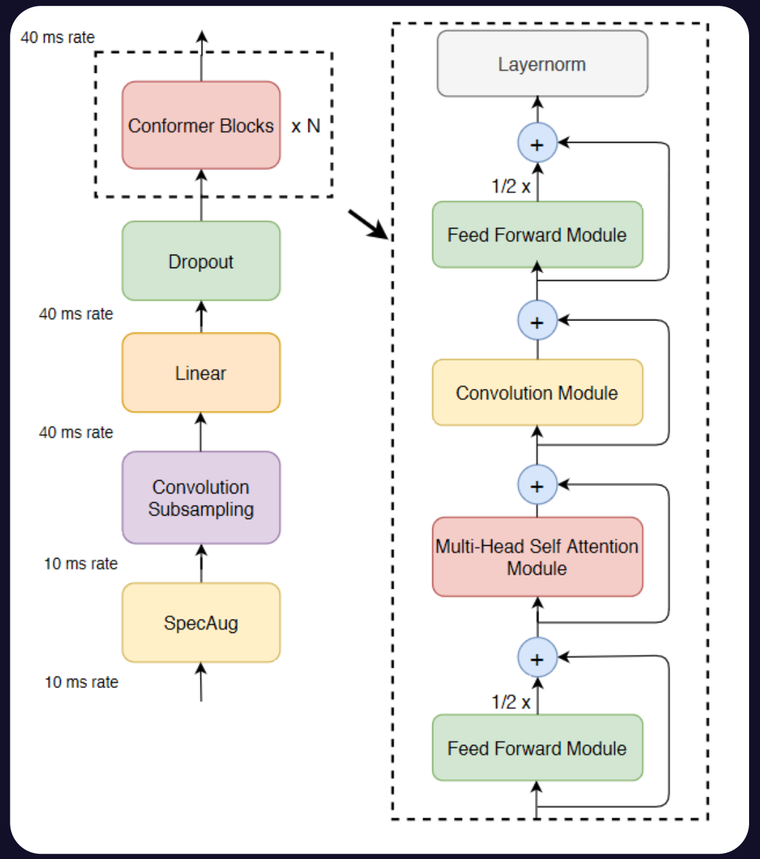
Model Attempt 2: Conformer-CtC

Realizing the limitations of our approach with Wav2Vec2 and FAdam, we decided to pivot towards the Conformer-CTC models.

These models promised better performance and stability, particularly for the tasks at hand.

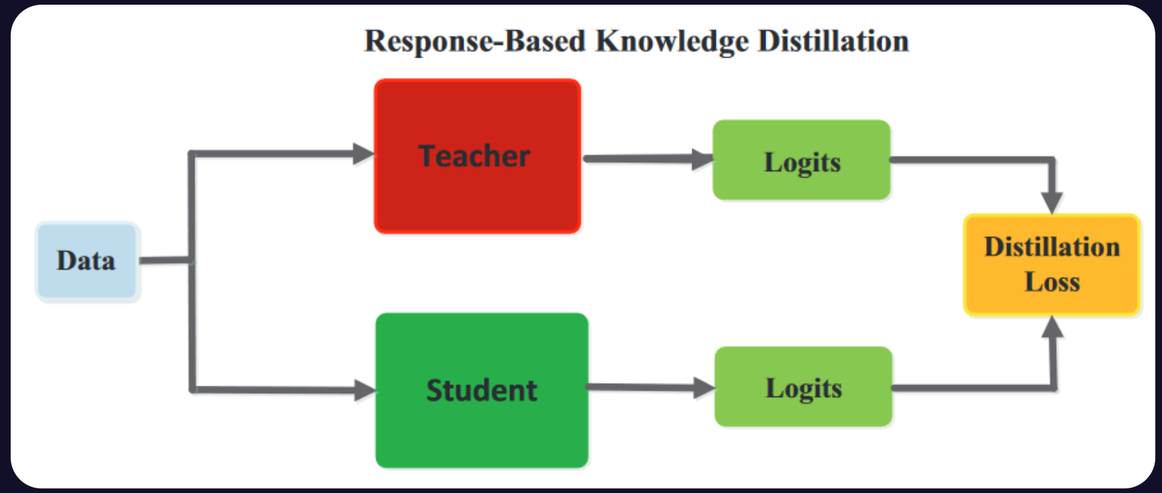
Conformer and Fast Conformer models currently represent the state-of-the-art in speech recognition.

The Conformer-CTC model combines the strengths of CNNs, transformers, and CTC for effective speech recognition tasks.



Knowledge Distillation Model Attempt

- Knowledge distillation is a process where a smaller model (the student) learns to mimic the behavior of a larger model (the teacher).
- Despite its potential benefits, we encountered difficulties with mismatched decoder lengths between Wav2Vec2 and Conformer-CTC, which disrupted the distillation process.
- A potential solution to this problem could be adjusting the transformer heads to match the output lengths of both models



Final model: Conformer-CTC

Model

- Conformer-CTC

Training Parameters

- Number of epochs: 45
- Tokenizer: SPE Unigram
- Learning rate: Warmup till 10k steps then decay
- Batch size: 16
- Optimizer: FAdam
- Loss function: Connectionist Temporal Classification loss (CTC)

Data Splitting

- Training set: Full train dataset
- Validation set: Adapt dataset

Training Procedure

- Train on the full dataset.
- Monitoring validation loss and accuracy during training.
- Fine-tune on adapt dataset for few epochs with 15% validation.

Data Augmentations: SpecAugment

- **Time Warping**: Time warping shifts the spectrogram in the time direction. This simulates slight variations in speaking speed and timing.
- **Frequency Masking**: one or more ranges of frequencies are masked (set to zero) randomly. This means certain frequency bands are removed. This helps the model become more invariant to different acoustic conditions.
- **Time Masking**: Similar to frequency masking, time masking involves masking out one or more time segments of the spectrogram.

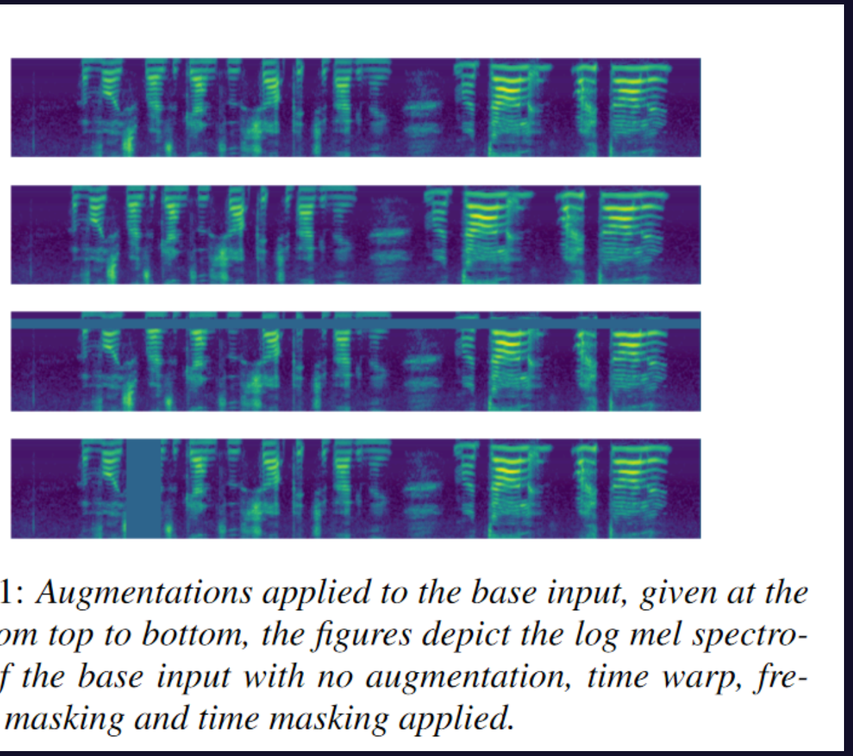


Figure 1: *Augmentations applied to the base input, given at the top. From top to bottom, the figures depict the log mel spectrogram of the base input with no augmentation, time warp, frequency masking and time masking applied.*

Unigram Tokenizer

Utilizes a subword segmentation algorithm based on a unigram language model. It aims to find the most likely subword units given the training data. We used the unigram with vocabulary size of 128 for the following reasons:

- **Reduced Vocabulary Size**: By breaking down words into smaller units, it significantly reduces the vocabulary size. This makes the model more efficient and effective in processing and understanding the language.
- **Better Generalization**: Smaller subword units can be recombined to form new words that the model has not explicitly seen during training. This improves the model's ability to generalize to new, unseen words.

Tokenizer

- Added **<fill> <overlap> <laugh>** tokens that was present in the transcriptions.
- Insure transcripts with these tokens are removed from the corpus used to build the tokenizer so the enligh letters are not part of the model vocabulary
- High learning rate set by default caused issues with all tokenizers except unigram

Unigram

['<unk>', 'ـف_', 'ج', 'ـب_', 'م', 'ش', 'أ', 'ـف', 'ح', 'ق', 'و', 'ـك', 'س', 'ع', 'ن', 'ب', 'و', 'د', 'ـ_', 'ـال_', 'ت', 'م', 'ل', 'ر', 'ه', 'ي', 'ـ_', 'ي', 'ص', 'ـل_', 'ـت_', 'خ', 'نا', 'ظ', 'ـإن_', 'ز', 'ين', 'ـع', 'يه', 'ـف', 'ض', 'ول', 'ال', 'ـي', 'ـمن', 'ات', 'ـح', 'ان', 'ـيعني', 'ـأنا', 'ـالم', 'ـ_', 'ـإ', 'ة', 'ـخ', 'ـمع', 'ـبت', 'ى', 'ـمش', 'ـالأ', 'ـعلى', 'ـش', 'ـبي', 'ده', 'ذ', 'وا', 'ـن', 'أ', 'ـمس', 'لا', 'غ', 'ها', 'ـق', 'ـك', 'ـاللي', 'ـ_', 'ج', 'بقى', 'ـأو', 'ـهو', 'ـكان', 'إ', 'ما', 'ـكل', 'ـعن', 'ت', 'هي', 'ـط', 'ـإم', 'ـر', 'ظ', 'ـلأ', 'ـإيه', 'ـدي', 'ـبال', 'ئ', 'ـكده', 'ـص', 'اء', 'ء', 'ؤ', 'ـحاجه', 'ـجدا', 'ـإحنا', '', '>', 'l', 'f', 'i', 'g', 'h', 'u', 'e', 'o', 'p', 'r', 'v', 'ـ', 'ق', '?', '÷', '،', 'a', '<', '=', '[', 'n', 'ج', 'أ', '١٠', 'ـ', 'آ']

Char

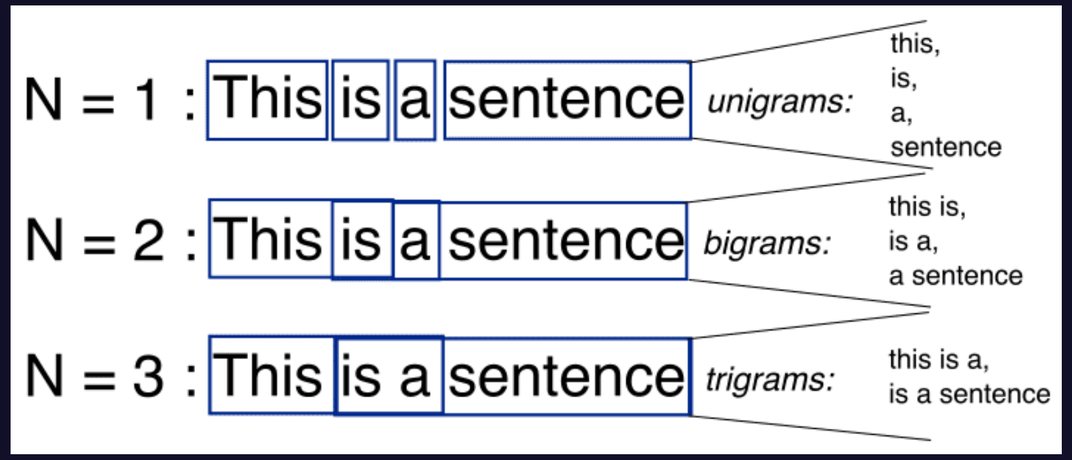
[' ', '<unk>', '<fill>', '<overlap>', '<laugh>', '_', 'ـ', 'ج', 'إ', 'ش', 'ق', 'ح', 'س', 'أ', 'ف', 'ك', 'د', 'ر', 'ع', 'ب', 'ت', 'و', 'ه', 'م', 'ن', 'ي', 'ل', 'ا', 'چ', 'آ', '١٠', 'ـ', '،', 'ق', 'ـ', 'أ', 'ؤ', 'ئ', 'ظ', 'ء', 'ث', 'ذ', 'غ', 'ة', 'ض', 'ز', 'ى', 'خ', 'ط', 'ص']

Language Model: N-gram

- We attempted to build a n-gram language model using the train and adapt transcripts. Also, tried using Egyptian Datasets Collection which is composed of 2.5 million rows after cleaning and removing emojis.

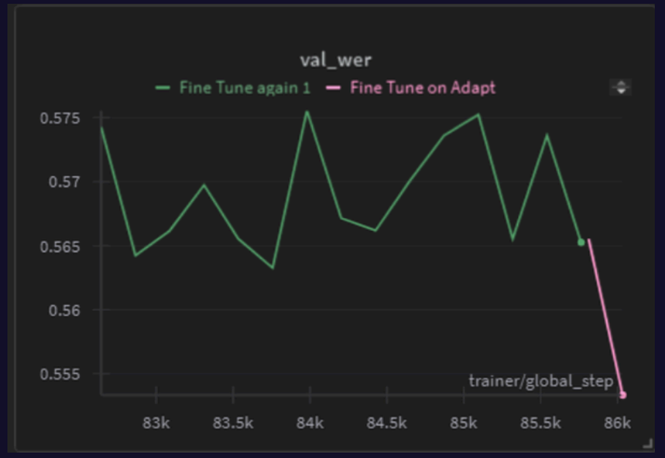
- Beam search is used in n-grams to avoid repeated phrases and improve the diversity and fluency of the generated text.

- We created multiple n-grams (3 to 6) but the results was worse and not as expected, as the beam search parameters needed extensive trial and error, so it was not included in the final model.

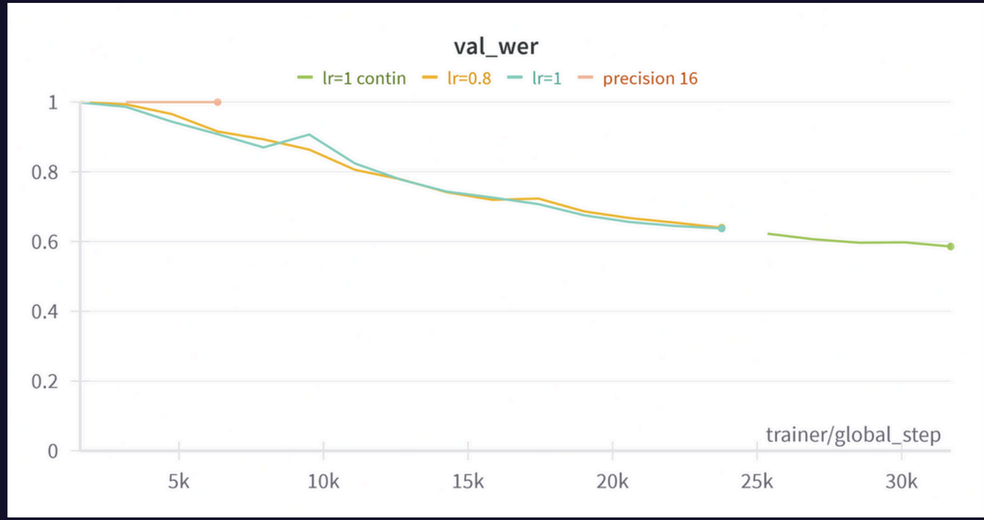


Results

The fluctuations in the wer are due to the high learning rate set by default, which we discovered late. However, this the model still achieved the best results and was used for the final submission that achieved 11.994785 mld



This demonstrates the char tokenizer results after tuning the learning rate and training for a 15 epochs



THANK YOU