Applying wav2vec2 for Speech Recognition on Bengali Common Voices Dataset

1st H.A.Z Sameen Shahgir
Undergraduate, Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh
sameen2080@gmail.com

2nd Khondker Salman Sayeed
Undergraduate, Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh
salmankhondker1998@gmail.com

3rd Abdullah Al Fahad
Undergraduate, Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh
alfahadarc@gmail.com

4th Somonindro Roy
Undergraduate, Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh
somoroy1290@gmail.com

***Abstract*—Speech is inherently continuous, where discrete words, phonemes and other units are not clearly segmented. Meta's general pre-trained model *wav2vec-2.0* is trained on the basic units of speech that transcend the context of a specific language. We have fine tuned *wav2vec 2.0* to the Bengali language by training it on the *Bengali Common Voice Speech Dataset*. After training over a test set of size 36919 mp3 files in 71 epochs, we achieved a training loss of 0.3172 and WER of 0.2524 on a validation set of size 7747. Using a 5-gram language model, we achieved a Levenshtein Mean Distance Score of 2.60753 on a hidden test set of size 7747.**

I. Introduction

Speech recognition and transcription is one of the quintessential applications of Machine Learning, with every advancement having wide reaching effects on society as a whole. English speech recognition in particular is at the stage of commercial viability, with products like Alexa, Google Assistant and Siri becoming household names. Unfortunately, the state of the art for Bengali speech recognition is lagging behind, especially considering that it is one of the most widely spoken languages in the world. Several factors such as the lack of commercial incentive, geological clustering of native speakers and a relatively young Bengali IT industry have caused this. Fortunately, new attention based models [8] such as Transformer have significantly reduced the data and compute requirements of previous RNN solutions.

Meta's *wav2vec 2.0* [3] is such an unsupervised model that is trained with a focus to learn the structure of languages. This single model is trained on multiple languages at the same time. Because of this, performance in low resource languages are benefited by training on it's sibling languages. Therefore, *wav2vec 2.0* being trained on South Asian languages like Hindi, Nepali, Urdu, Telegu indirectly benefits speech unit identification of Bengali as well.

Recently, an addition to the repository of Bengali speech transcription was made by *Bengali Common Voice Speech Dataset* [1]. This dataset allows us to fine tune Meta's pre-trained *wav2vec 2.0* to Bengali to allow better transcription specific to Bengali. In this paper we showcase the training pipeline used to train *wav2vec 2.0* to *Bengali Common Voice Speech Dataset* to achieve a considerable Word Error Rate (WER).

II. Methodology

A. Pre-processing

The train split of *Bengali Common Voice Speech Dataset* comprises of 206,951 mp3 files with their corresponding Bengali transcription along with some meta-data such as up-votes, downvotes, gender etc. On quick analysis, the following metrics were found.

Criterion	Count
$upvotes > downvotes$	37405
$upvotes < downvotes$	5536
$upvotes = downvotes = 0$	161380

Since 5536 out of 42941 ($\sim$13 percent) voted data was unreliable, we opted to train using only the subset with $upvotes > downvotes$.

The mp3 files were then sampled at 16kHz and characters in the label strings were cast to integer hashes using the a vocabulary dictionary forked from *arijitx/wav2vec2-xls-r-300m-bengali* [6].

The Transformers implementation from the Transformers library pads each array to match the longest array in the same batch. For faster training, we removed the starting and ending portions of the each sound array if the value at that index was less than some fraction of the maximum value in that array. After some testing, the cutoff threshold $x < \frac{max(array)}{30}$ was chosen.

Finally, only sound files between 1 and 9 seconds were chosen for training, yielding a final train set of length 36919.

B. Training

For training we used the pytorch [7] based implementation of Transformer model provided and maintained by huggingface.co [4]. We fine-tuned the pretrained *facebook/wav2vec2-large-xlsr-53* [3] which was trained to on unlabelled multilingual speech and intended for further downstream training on labelled data.

First Phase Training was done for a total of 71 epochs, totalling to around 100 hours of training on Kaggle provided Nvidia K80 GPU Kernels. The AdamW Optimizer was used starting with learning rate 5×10^{-4} and weight delay of 2.5×10^{-6}. These values were decided on after some preliminary attempts which either failed to converge with larger hyper-parameters or were slow to show appreciable convergence with lower rates.

Around 0.25 WER ($\sim$ Epoch 55), we noticed the training and evaluation metrics plateauing considerably. On 7747 hidden test data, we achieved a Levenshtein Mean Distance Score of 2.64446. Speech recognition tasks often hinge on exposure to a large vocabulary and since we initially limited ourselves to a subset of the entire dataset, we used a portion of the validation set for further training.

Exposure to new data is not sound in principle due to older weights decaying without continued exposure to the original dataset. For this reason, we merged training and validation datasets and applied a 90-10 split. With a weight decay of 2.5×10^{-9} and learning rate 5×10^{-6}, we trained for a further 10 epochs. This produced a noticeable improvement to the Levenshtein Mean Distance Score, now 2.60753 (down from 2.6446).

C. Post-processing

Three separate post processing functions were used to generate the final prediction on the hidden test set, namely n-gram Language Model decoding, unicode normalization and appending a common character to the end of sentences.

Our Language Model of choice was forked from *arijitx/wav2vec2-xls-r-300m-bengali* [6] generated from IndicCorp language corpus [5]. The language model improved spelling accuracy and the Levenshtein Mean Distance Score decreased from 3.30648 to 2.72243 once applied.

Unicode Normalization using *bnunicodenormalizer* [2] and appending the ”devanagari danda” [Unicode#2404] to output sentences also resulted in slight improvements.

III. Results

After a total of about 77 epochs of training, in two phases – our model settled on a Levenshtein Mean Distance Score of 2.60753. The training error and evaluation word error metric gradually decreased through out the training process. Our last submission had a CTC training error of 0.3172, and an evaluation WER of 0.2524.

From figure 2 we can see that WER has plateaued at around 77 epochs. Whereas training loss, as in figure 1, could have been pushed further down.

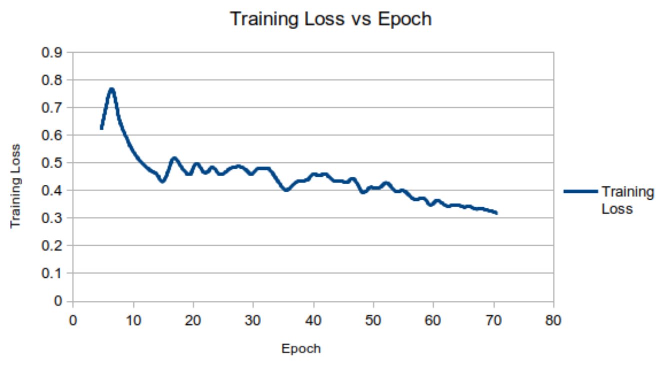

Fig. 1. Traning Loss vs Epoch

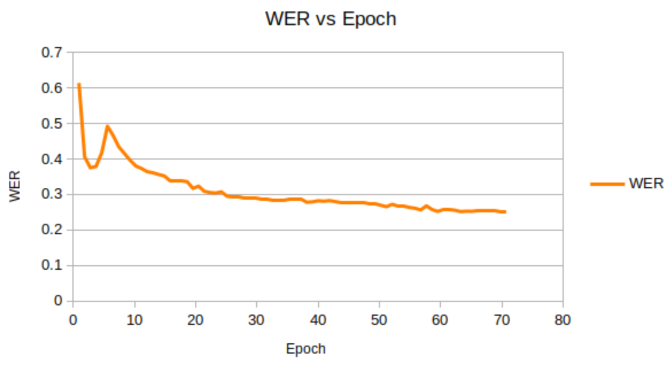

Fig. 2. WER vs Epoch

IV. Discussion

While training it was observed there that the training loss did not correspond strongly to the Word Error Rate. Since WER is the final evaluation metric, epochs in the later stages of training sometimes resulted in slight increase in training loss. Unlike other neural network based models, the risk of over-fitting appears to be low in the Transformer model.

The model was also able to adapt to exposure to new data well and it did not result in any sudden spike in evaluation metrics. On the second dataset, we observed that even with training loss and WER plateauing, further training resulted in a slightly improved Levenshtein Mean Distance Score. This is likely due to the exposure to new vocabulary which ultimately improves performance on previously unseen data.

It is our belief that we didn’t push the limits of the wav2vec2 Transformer model since it achieved a best case WER of 0.18 [3] without using a language model. With more labelled data and better compute, it is highly likely a better result on Bengali speech recognition can be achieved using this model.

V. Conclusion

In this work, we apply the pre-trained wav2vec2.0 model to transcribe the Bengali Common Voices Dataset, achieving a final WER of 0.2524, which is in the range (0.18-0.33) achieved by wav2vec2 on the Librispeech Dataset [3]. This demonstrates the viability of the wav2vec2 model for Bengali Speech Recognition. Furthermore, this result was achieved using only 17.84% of the Bengali Common Voices Dataset which attests to the excellent quality of the dataset. A larger subset of the the dataset can possibly achieve even better results with further training.

REFERENCES

[1] Samiul Alam, Asif Sushmit, Zaowad Abdullah, Shahrin Nakkhatra, MD. Nazmuddoha Ansary, Syed Mobassir Hossen, Sazia Morshed Mehnaz, Tahsin Reasat, and Ahmed Imtiaz Humayun. Bengali common voice speech dataset for automatic speech recognition, 2022.

[2] MD. Nazmuddoha Ansary. Bengali Unicode Normalizer. https://github.com/mnansary/bnUnicodeNormalizer, 2022.

[3] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. facebook/wav2vec2-large-xlsr-53. https://huggingface.co/facebook/wav2vec2-large-xlsr-53, 2022.

[4] huggingface. Huggingface Transformers. https://huggingface.co/docs/transformers/index, 2022.

[5] Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In *Findings of EMNLP*, 2020.

[6] Arijit Mukherjee. arijitx/wav2vec2-xls-r-300m-bengali. https://huggingface.co/arijitx/wav2vec2-xls-r-300m-bengali, 2022.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.