Vietnamese Automatic Speech Recognition

Evan Williams
Carnegie Mellon University
emwillia@cmu.edu

Charity King
Carnegie Mellon University
csking@andrew.cmu.edu

Abstract

End-to-End (E2E) models have become popular approaches for speech recognition and other speech processing tasks. The performance of E2E models on automatic speech recognition (ASR) tasks has greatly improved specifically in regard to high-density languages with large amounts of data. This project evaluates the use of the E2E speech processing toolkit ESPnet package against Vietnamese, a small data set to better understand challenges and performance-enhancing techniques against smaller data sets.

1 Introduction

While the performance of Automatic Speech Recognition (ASR) models has made substantial progress in recent years, these improvements have largely been restricted to high-resource languages, which often require tens-of-thousands of hours of annotated speech data (Khare et al., 2021). For the majority of languages in the world, large annotated speech datasets are not available. As a result, there's widespread interest in improving the performance of E2E ASR systems on low-resource languages.

Vietnamese is similar to Chinese in that it is widely considered a 'monosyllabic' language; a language where words are one syllable long. Vietnamese is comprised of a large number of free forms, in addition to the tradition of writing and printing each syllable separated by a space (Thompson, 1963). Experimenting with the tokenization of these characters will be a large part of our hyperparamter tuning.

Additionally, Vietnamese is a challenging language from an ASR perspective. Speakers in Vietnamese often intentionally mispronounce words, e.g., pronouncing 'ɹ' like a 'z', because it sounds better. These substitutions are very common in songs, but also appear in casual speech as well. Regional dialects add additional complexity, as emphasis and accents and tones can vary substantially. There are also many filler words/ filler-syllables in the language.

In this paper, we explore the performance of a baseline ESPnet model recipe run on a Vietnamese Mozilla Common Voice dataset (Watanabe et al., 2018) to better understand the impact of hyperparameter tuning in improving the performance of our model against the baseline, given a limited dataset.

2 Dataset

We use the Mozilla Common Voice Version 5.1 Vietnamese dataset. This corpus is a very small dataset of approximately 15 minutes of audio from 14 different voices. The dataset contains predominantly younger voices, with, 74% of the self-reported speakers between 19-29 years old, 9% between 30-39 years old, and only 2% self-identified as being under 19 years old. Additionally, the data skews on the gender line with 69% of voices as male, 16% female. All models were trained using GPU available on Google Colab.

3 Baseline Model

For our baseline model, we use a built-in ESPNet recipe. We restrict min and max wave duration to 0.1 and 20 respectively. We use a unigram byte pair encoding (bpe) with a vocabulary length of 150. The use of BPEs as tokenizers have recently become popularized in contextual natural language processing models such as BERT and GPT-2 due to the subword tokenization and character and word-level hybrid representations suitable for large corpora (Sennrich et al.) .

To augment the data, we use speed perturbations of 0.9, 1.0, and 1.1 as recommended by (Ko et al., 2015). We use a conformer architecture with 12 encoder blocks, 4 attention heads in each block, a dropout of 0.1, and a swish activation function. The baseline conformer has 6 decoder blocks, hybrid connectionist temporal classification (CTC) and attention weighting.

Our baseline uses an Adam optimizer with a Noam learning-rate scheduler, a starting learning rate factor of 4.0, and 25k warmup steps. The Noam learning rate scheduler increases learning rate during warmup and then decreases the learning rate during training, proportional to the inverse square root of the step number and scaled by the inverse square root of the dimensionality of the model (Vaswani et al., 2017). We run the model for a total of 50 epochs.

4 Evaluation

We evaluate each of our models using both the Character Error Rate (CER) and Word Error Rate (WER). The Error rate is sum of the number of deletions, insertions, and substitutions, divided by the number of characters in the reference. Due to the low-accuracy of our baseline which was around 12% on our validation data set, both of these evaluation metrics are necessary to understand the trade-offs that occur with hyperparameter tuning in regards to both WER and CER.

5 Experiments

We explore the impact of modifying the initial learning rate, tokenizer vocabulary count, and the number of encoder and decoder blocks present in the model.

5.1 Learning Rate

We explored several different initial Learning Rates for the Noam learning-rate scheduler. We found initial learning rates of 3.0 and 5.0 yielded the lowest CER, whereas an initial learning rate of 1.0 yielded the lowest WER. Results are displayed in Table 1.

Initial LR	**WER**	**CER**
0.01	618.1	589.6
1.0	**100**	88.3
2.0	103.6	80.8
3.0	104.3	**79.9**
4.0 (Baseline)	103.6	80.7
5.0	104.3	**79.9**
6.0	101.7	83.7

Table 1: CER and WER at different initial Learning Rates for the Noam learning-rate scheduler

5.2 BPE Tokenizer

We first explored using character and word tokenizers instead of the baseline BPE tokenizer. We found that the BPE tokenizer yielded the lowest CER and the character tokenizer yielded the highest CER as can be seen in Table 2. This was surprising to us, as we're using Character Error rate as our evaluation metric.

Tokenizer	**WER**	**CER**
Character	254.5	106.3
Word	**93.3**	95.7
BPE (baseline)	103.6	**80.7**

Table 2: WER and CER of Character and Word tokenizers compared to BPE baseline

We then explored the impact of BPE Vocabulary size on CER. We tried modifying the BPE vocabulary count by increments of 25 between 100 and 200 inclusive. We found that a vocab count of 125 yielded the best CER for our model and a vocab of 200 yielded the best WER as can be seen in Table 3. We also found this somewhat surprising, as we expected a larger vocabulary would allow the model to better learn more complex patterns. Additionally due to similarities between Chinese and Vietnamese, we hypothesized that a high BPE similar to Chinese where training utilizes a BPE = 300 would result in higher performance, but there is no clear monotonic relationship between BPE and consistent WER and CER improvement.

BPE Vocab	**WER**	**CER**
100	240	93.6
125	110.2	**78.4**
150 (baseline)	103.6	80.7
175	105.2	78.8
200	**100.2**	86.1

Table 3: WER and CER for BPE Vocab count

5.3 Encoder and Decoder Blocks

Finally, we explored the impact of changing the number of encoder and decoder blocks in the baseline model. Interestingly, as the number of encoder and decoder blocks increased, WER increases, but CER decreases as shown in Table 4. We found this to be our strangest finding, as the definitions of WER and CER are similar, we would imagine them to move in the same direction as the model becomes larger. Additionally, we were surprised WER increased because more weights should allow the model to better capture the complex patterns that the model needs to learn.

Enc-Dec-Blocks	WER	CER
10E5D	100	84.3
12E6D (baseline)	103.6	80.7
14E7D	104.3	**78.5**

Table 4: WER and CER vs. number of encoders and number of decoders

6 Results

Finally, we train two models using the hyperparameters that yielded the lowest WER and CER scores in each experiment. For the WER-optimized model, we use an initial learning rate of 1.0, a word tokenizer with a vocab length of 200, 10 encoder blocks, and 5 decoder blocks. For the CER-optimized moel, we use an initial learning rate of 5.0, a BPE tokenizer with a vocabulary length of 25, 14 encoder blocks, and 7 decoder blocks. Both of these models improve WER and CER over the baseline models, but they do not surpass the scores achieved in some of our experiments. Our WER-optimized model yielded a WER of 99.0 and our CER-optimized model yielded a CER of 79.7 as can be seen in Table 5.

	WER	CER
WER-opitimzed	**99.0**	95.2
CER-opitimzed	106.0	**79.7**

Table 5: Final models trained using the best hyperparameters we found for WER and CER models

The lowest WER score we achieved was 93.3, using the baseline architecture and the word tokenizer. The lowest CER we achieved was 78.5, using baseline hyperparameters with 14 encoders and 7 decoders.

7 Conclusion

By tuning the baseline's hyperparameters in ESPNET, we were able to improve WER by 10.2 points (11% improvement) over the baseline and improve CER by 2.2 points (2.8% improvement). Our final models only achieved 4.5 WER and 1.0 CER improvements respectively. Given the non-linear interactions between hyperparameters, in future work we should not hold hyperparameters constant when tuning.

References

Shreya Khare, Ashish Mittal, Anuj Diwan, Sunita Sarawagi, Preethi Jyothi, and Samarth Bharadwaj. 2021. Low resource asr: The surprising effectiveness of high resource transliteration. *Proc. Interspeech 2021*, pages 1529–1533.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *Sixteenth annual conference of the international speech communication association*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics.

Laurence C. Thompson. 1963. The problem of the word in vietnamese. 19(1):39–52.

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

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. 2018. ESPnet: End-to-end speech processing toolkit. In *Proceedings of Interspeech*, pages 2207–2211.